A Guide on Data Analysis

Mike Nguyen

2025-02-18

Contents

Preface **11**
How to cite this book . . . 12
More books . . . 13

1 Introduction **15**
1.1 General Recommendations . . . 16

2 Prerequisites **23**
2.1 Matrix Theory . . . 23
2.2 Probability Theory . . . 35
2.3 General Math . . . 79
2.4 Data Import/Export . . . 91
2.5 Data Manipulation . . . 97

I. BASIC **111**

3 Descriptive Statistics **111**
3.1 Numerical Measures . . . 111
3.2 Graphical Measures . . . 114
3.3 Normality Assessment . . . 121
3.4 Bivariate Statistics . . . 137

4 Basic Statistical Inference **169**
4.1 Hypothesis Testing Framework . . . 170
4.2 Key Concepts and Definitions . . . 178
4.3 One-Sample Inference . . . 180
4.4 Two-Sample Inference . . . 201
4.5 Categorical Data Analysis . . . 228
4.6 Divergence Metrics and Tests for Comparing Distributions . . . 269

II. REGRESSION **301**

5 Linear Regression **301**
5.1 Ordinary Least Squares . . . 306
5.2 Generalized Least Squares . . . 415
5.3 Maximum Likelihood . . . 434
5.4 Penalized (Regularized) Estimators . . . 449
5.5 Robust Estimators . . . 458
5.6 Partial Least Squares . . . 482

6 Non-Linear Regression **491**
6.1 Inference . . . 493
6.2 Non-linear Least Squares Estimation . . . 498
6.3 Practical Considerations . . . 609
6.4 Application . . . 619

7 Generalized Linear Models **639**
7.1 Logistic Regression . . . 640
7.2 Probit Regression . . . 658
7.3 Binomial Regression . . . 662
7.4 Poisson Regression . . . 667
7.5 Negative Binomial Regression . . . 674
7.6 Quasi-Poisson Regression . . . 680
7.7 Multinomial Logistic Regression . . . 686
7.8 Generalization of Generalized Linear Models . . . 697

8 Linear Mixed Models **731**
8.1 Dependent Data . . . 731
8.2 Estimation in Linear Mixed Models . . . 748
8.3 Inference in Linear Mixed Models . . . 761
8.4 Information Criteria for Model Selection . . . 765
8.5 Split-Plot Designs . . . 771
8.6 Repeated Measures in Mixed Models . . . 779
8.7 Unbalanced or Unequally Spaced Data . . . 782
8.8 Application: Mixed Models in Practice . . . 785

9 Nonlinear and Generalized Linear Mixed Models **801**
9.1 Nonlinear Mixed Models . . . 801
9.2 Generalized Linear Mixed Models . . . 802
9.3 Relationship Between NLMMs and GLMMs . . . 803
9.4 Marginal Properties of GLMMs . . . 803
9.5 Estimation in Nonlinear and Generalized Linear Mixed Models . 805
9.6 Application: Nonlinear and Generalized Linear Mixed Models . . 816
9.7 Summary . . . 838

10 Nonparametric Regression **839**
10.1 Why Nonparametric? . . . 840

10.2 Basic Concepts in Nonparametric Estimation 843
10.3 Kernel Regression 844
10.4 Local Polynomial Regression 862
10.5 Smoothing Splines 875
10.6 Confidence Intervals in Nonparametric Regression 882
10.7 Generalized Additive Models 884
10.8 Regression Trees and Random Forests 895
10.9 Wavelet Regression 909
10.10 Multivariate Nonparametric Regression 917
10.11 Conclusion: The Evolving Landscape of Regression Analysis . . . 928

III. RAMIFICATIONS 933

11 Data 933
11.1 Data Types 933
11.2 Cross-Sectional Data 937
11.3 Time Series Data 938
11.4 Repeated Cross-Sectional Data 953
11.5 Panel Data 957
11.6 Choosing the Right Type of Data 1018
11.7 Data Quality and Ethical Considerations 1018

12 Variable Transformation 1021
12.1 Continuous Variables 1021
12.2 Categorical Variables 1040

13 Imputation (Missing Data) 1051
13.1 Introduction to Missing Data 1051
13.2 Theoretical Foundations of Missing Data 1055
13.3 Diagnosing the Missing Data Mechanism 1061
13.4 Methods for Handling Missing Data 1067
13.5 Evaluation of Imputation Methods 1102
13.6 Criteria for Choosing an Effective Approach 1104
13.7 Challenges and Ethical Considerations 1105
13.8 Emerging Trends in Missing Data Handling 1107
13.9 Application of Imputation 1110

14 Model Specification Tests 1127
14.1 Nested Model Tests 1127
14.2 Non-Nested Model Tests 1135
14.3 Heteroskedasticity Tests 1146
14.4 Functional Form Tests 1156
14.5 Autocorrelation Tests 1163
14.6 Multicollinearity Diagnostics 1171

15 Variable Selection **1181**
15.1 Filter Methods (Statistical Criteria, Model-Agnostic) 1183
15.2 Wrapper Methods (Model-Based Subset Evaluation) 1203
15.3 Embedded Methods (Integrated into Model Training) 1221
15.4 Summary Table 1225

16 Hypothesis Testing **1227**
16.1 Null Hypothesis Significance Testing 1227
16.2 Two One-Sided Tests Equivalence Testing 1252
16.3 False Discovery Rate 1257
16.4 Comparison of Testing Frameworks 1265

17 Marginal Effects **1267**
17.1 Definition of Marginal Effects 1267
17.2 Marginal Effects in Different Contexts 1270
17.3 Marginal Effects Interpretation 1271
17.4 Delta Method 1271
17.5 Comparison: Delta Method vs. Alternative Approaches 1272
17.6 Types of Marginal Effect 1277
17.7 Packages for Marginal Effects 1286

18 Moderation **1293**
18.1 Types of Moderation Analyses 1293
18.2 Key Terminology 1293
18.3 Moderation Model 1294
18.4 Types of Interactions 1294
18.5 Three-Way Interactions 1294
18.6 Additional Resources 1294
18.7 Application 1295

19 Mediation **1327**
19.1 Traditional Approach 1327
19.2 Causal Inference Approach to Mediation 1345

20 Prediction and Estimation **1357**
20.1 Conceptual Framing 1357
20.2 Mathematical Setup 1358
20.3 Prediction in Detail 1360
20.4 Parameter Estimation and Causal Inference 1364
20.5 Causation versus Prediction 1366
20.6 Illustrative Equations and Mathematical Contrasts 1368
20.7 Extended Mathematical Points 1370
20.8 Putting It All Together: Comparing Objectives 1372
20.9 Conclusion 1372

IV. CAUSAL INFERENCE 1377

21 Causal Inference **1377**
21.1 The Ladder of Causation 1379
21.2 The Formal Notation of Causality 1380
21.3 The 7 Tools of Structural Causal Models 1380
21.4 Simpson's Paradox 1381
21.5 Additional Resources 1388
21.6 Experimental vs. Quasi-Experimental Designs 1389
21.7 Hierarchical Ordering of Causal Tools 1389
21.8 Types of Validity in Research 1390
21.9 Types of Subjects in a Treatment Setting 1395
21.10 Types of Treatment Effects 1397

A. EXPERIMENTAL DESIGN 1421

22 Experimental Design **1421**
22.1 Principles of Experimental Design 1421
22.2 The Gold Standard: Randomized Controlled Trials 1422
22.3 Selection Problem 1422
22.4 Classical Experimental Designs 1426
22.5 Advanced Experimental Designs 1434
22.6 Emerging Research 1445

23 Sampling **1453**
23.1 Population and Sample 1453
23.2 Sampling Techniques 1453
23.3 Unequal Probability Sampling 1458
23.4 Balanced Sampling 1459
23.5 Sample Size Determination 1462

24 Analysis of Variance **1463**
24.1 Completely Randomized Design 1464
24.2 Single-Factor Fixed Effects Model 1465
24.3 Linear Model Interpretation of ANOVA 1467
24.4 Expected Mean Square for Treatments 1469
24.5 F-Test for Equality of Treatment Means 1469
24.6 Nonparametric ANOVA 1503
24.7 Sample Size Planning for ANOVA 1505
24.8 Randomized Block Designs 1507
24.9 Nested Designs 1513
24.10 Single Factor Covariance Model 1517

25 Multivariate Methods **1521**
25.1 MANOVA 1544

25.2 Principal Components 1562
25.3 Factor Analysis 1573
25.4 Discriminant Analysis 1589

B. QUASI-EXPERIMENTAL DESIGN 1615

26 Quasi-experimental 1615
26.1 Natural Experiments 1616

27 Regression Discontinuity 1621
27.1 Estimation and Inference 1624
27.2 Specification Checks 1625
27.3 Fuzzy RD Design 1640
27.4 Regression Kink Design 1641
27.5 Multi-cutoff 1642
27.6 Multi-score 1642
27.7 Steps for Sharp RD 1642
27.8 Steps for Fuzzy RD 1642
27.9 Steps for RDiT (Regression Discontinuity in Time) 1643
27.10 Evaluation of an RD 1646
27.11 Applications 1648

28 Synthetic Difference-in-Differences 1655
28.1 Understanding 1656
28.2 Application 1661

29 Difference-in-differences 1673
29.1 Visualization 1674
29.2 Simple Dif-n-dif 1677
29.3 Notes 1681
29.4 Standard Errors 1690
29.5 Examples 1690
29.6 One Difference 1695
29.7 Two-way Fixed-effects 1696
29.8 Multiple periods and variation in treatment timing 1702
29.9 Staggered Dif-n-dif 1705
29.10 Multiple Treatments 1783
29.11 Mediation Under DiD 1784
29.12 Assumptions 1784

30 Changes-in-Changes 1793
30.1 Application 1795

31 Synthetic Control 1807
31.1 Applications 1812
31.2 Augmented Synthetic Control Method 1828

31.3 Synthetic Control with Staggered Adoption 1828
31.4 Bayesian Synthetic Control 1828
31.5 Generalized Synthetic Control 1828
31.6 Other Advances 1828

32 Event Studies **1829**
32.1 Other Issues 1833
32.2 Testing 1837
32.3 Sample 1839
32.4 Biases 1844
32.5 Long-run event studies 1845
32.6 Aggregation 1851
32.7 Heterogeneity in the event effect 1852
32.8 Expected Return Calculation 1853
32.9 Application 1855

33 Instrumental Variables **1861**
33.1 Framework 1864
33.2 Estimation 1867
33.3 Inference 1872
33.4 Testing Assumptions 1880
33.5 Negative R^2 1891
33.6 Treatment Intensity 1892
33.7 Control Function 1893
33.8 New Advances 1900

34 Matching Methods **1903**
34.1 Selection on Observables 1915
34.2 Selection on Unobservables 1935

35 Interrupted Time Series **1945**
35.1 Autocorrelation 1951
35.2 Multiple Groups 1953

C. OTHER CONCERNS **1957**

36 Endogeneity **1957**
36.1 Endogenous Treatment 1959
36.2 Endogenous Sample Selection 1988
36.3 Implications for Causal Inference 1989

37 Other Biases **2009**
37.1 Aggregation Bias 2011
37.2 Contamination Bias 2019
37.3 Survivorship Bias 2019
37.4 Publication Bias 2022

38 Controls **2027**
38.1 Bad Controls . . . 2027
38.2 Good Controls . . . 2041
38.3 Neutral Controls . . . 2056
38.4 Choosing Controls . . . 2066

39 Directed Acyclic Graph **2069**
39.1 Basic Notations . . . 2069

V. MISCELLANEOUS **2073**

40 Report **2073**
40.1 One summary table . . . 2074
40.2 Model Comparison . . . 2076
40.3 Changes in an estimate . . . 2082
40.4 Standard Errors . . . 2083
40.5 Coefficient Uncertainty and Distribution . . . 2084
40.6 Descriptive Tables . . . 2085
40.7 Visualizations and Plots . . . 2087

41 Exploratory Data Analysis **2093**

42 Sensitivity Analysis/ Robustness Check **2095**
42.1 Specification curve . . . 2095
42.2 Coefficient stability . . . 2111
42.3 Omitted Variable Bias Quantification . . . 2112

43 Replication and Synthetic Data **2115**
43.1 The Replication Standard . . . 2115
43.2 Synthetic Data . . . 2118
43.3 Application . . . 2120

A Appendix **2147**
A.1 Git . . . 2147
A.2 Short-cut . . . 2149
A.3 Function short-cut . . . 2149
A.4 Citation . . . 2151
A.5 Install all necessary packages/libaries on your local machine . . . 2151

B Bookdown cheat sheet **2155**
B.1 Operation . . . 2155
B.2 Math Expression/ Syntax . . . 2156
B.3 Table . . . 2160

Preface

This book is an effort to simplify and demystify the complex process of data analysis, making it accessible to a wide audience. While I do not claim to be a professional statistician, econometrician, or data scientist, I firmly believe in the value of learning through teaching and practical application. Writing this book has been as much a learning journey for me as I hope it will be for you.

The intended audience includes those with little to no experience in statistics, econometrics, or data science, as well as individuals with a budding interest in these fields who are eager to deepen their knowledge. While my primary domain of interest is marketing, the principles and methods discussed in this book are universally applicable to any discipline that employs scientific methods or data analysis.

I hope this book provides a valuable starting point for aspiring statisticians, econometricians, and data scientists, empowering you to navigate the fascinating world of causal inference and data analysis with confidence.

How to cite this book

1. APA (7th edition):

Nguyen, M. (2020). *A Guide on Data Analysis*. Bookdown.

https://bookdown.org/mike/data_analysis/

2. MLA (8th edition):

Nguyen, Mike. *A Guide on Data Analysis*. Bookdown, 2020. **https://bookdown.org/mike/data_analysis/**

3. Chicago (17th edition):

Nguyen, Mike. 2020. *A Guide on Data Analysis*. Bookdown. **https://bookdown.org/mike/data_analysis/**

4. Harvard:

Nguyen, M. (2020) *A Guide on Data Analysis*. Bookdown. Available at: **https://bookdown.org/mike/data_analysis/**

```
@book{nguyen2020guide,
  title={A Guide on Data Analysis},
  author={Nguyen, Mike},
  year={2020},
  publisher={Bookdown},
```

```
  url={https://bookdown.org/mike/data_analysis/}
}
```

More books

More books by the author can be found here:

- Advanced Data Analysis: the second book in the data analysis series, which covers machine learning models (with a focus on prediction)
- Marketing Research
- Communication Theory

Chapter 1

Introduction

Since the turn of the century, we have witnessed remarkable advancements and innovations, particularly in statistics, information technology, computer science, and the rapidly emerging field of data science. However, one challenge of these developments is the overuse of **buzzwords** like *big data*, *machine learning*, and *deep learning*. While these terms are powerful in context, they can sometimes obscure the foundational principles underlying their application.

Every substantive field often has its own specialized metric subfield, such as:

- **Econometrics** in economics
- **Psychometrics** in psychology
- **Chemometrics** in chemistry
- **Sabermetrics** in sports analytics
- **Biostatistics** in public health and medicine

To the layperson, these disciplines are often grouped under broader terms like:

- **Data Science**
- **Applied Statistics**
- **Computational Social Science**

As exciting as it is to explore these new tools and techniques, I must admit that retaining these concepts can be challenging. For me, the most effective way to internalize and apply these ideas has been to document the data analysis process from start to finish.

With that in mind, let's dive in and explore the fascinating world of data analysis together.

1.1 General Recommendations

- The journey of mastering data analysis is fueled by practice and repetition. The more lines of code you write, the more functions you familiarize yourself with, and the more you experiment, the more enjoyable and rewarding this process becomes.
- Readers can approach this book in several ways:
  - **Focused Learning**: If you are interested in specific methods or tools, you can jump directly to the relevant section by navigating through the table of contents.
  - **Sequential Learning**: To follow a traditional path of data analysis, start with the Linear Regression section.
  - **Experimental Approach**: If you are interested in designing experiments and testing hypotheses, explore the [Analysis of Variance (ANOVA)] section.
- For those primarily interested in applications and less concerned with theoretical foundations, focus on the summary and application sections of each chapter.

- If a concept is unclear, consider researching the topic online. This book serves as a guide, and external resources like tutorials or articles can provide additional insights.
- To customize the code examples provided in this book, use R's built-in help functions. For instance:
  - To learn more about a specific function, type `help(function_name)` or `?function_name` in the R console.
  - For example, to find details about the `hist` function, type `?hist` or `help(hist)` in the console.
- Additionally, searching online is a powerful resource (e.g., Google, ChatGPT, etc.). Different practitioners often use various R packages to achieve similar results. For instance, if you need to create a histogram in R, a simple search like *"histogram in R"* will provide multiple approaches and examples.

By adopting these strategies, you can tailor your learning experience and maximize the value of this book.

Tools of statistics

- Probability Theory
- Mathematical Analysis
- Computer Science
- Numerical Analysis
- Database Management

Code Replication

This book was built with R version 4.4.2 (2024-10-31 ucrt) and the following packages:

package	version	source
abind	1.4-5	CRAN (R 4.2.0)
agridat	1.21	CRAN (R 4.2.3)
ape	5.7-1	CRAN (R 4.2.3)
assertthat	0.2.1	CRAN (R 4.2.3)
backports	1.4.1	CRAN (R 4.2.0)
bookdown	0.35	CRAN (R 4.2.3)
boot	1.3-28.1	CRAN (R 4.2.3)
broom	1.0.5	CRAN (R 4.2.3)
bslib	0.6.1	CRAN (R 4.2.3)
cachem	1.0.8	CRAN (R 4.2.3)
callr	3.7.3	CRAN (R 4.2.3)
car	3.1-2	CRAN (R 4.2.3)
carData	3.0-5	CRAN (R 4.2.3)
cellranger	1.1.0	CRAN (R 4.2.3)

package	version	source
cli	3.6.1	CRAN (R 4.2.3)
coda	0.19-4	CRAN (R 4.2.3)
colorspace	2.1-0	CRAN (R 4.2.3)
corpcor	1.6.10	CRAN (R 4.2.0)
crayon	1.5.2	CRAN (R 4.2.3)
cubature	2.1.0	CRAN (R 4.2.3)
curl	5.1.0	CRAN (R 4.2.3)
data.table	1.14.8	CRAN (R 4.2.3)
DBI	1.2.0	CRAN (R 4.2.3)
dbplyr	2.4.0	CRAN (R 4.2.3)
desc	1.4.3	CRAN (R 4.2.3)
devtools	2.4.5	CRAN (R 4.2.3)
digest	0.6.31	CRAN (R 4.2.3)
dplyr	1.1.2	CRAN (R 4.2.3)
ellipsis	0.3.2	CRAN (R 4.2.3)
evaluate	0.23	CRAN (R 4.2.3)
extrafont	0.19	CRAN (R 4.2.2)
extrafontdb	1.0	CRAN (R 4.2.0)
fansi	1.0.4	CRAN (R 4.2.3)
faraway	1.0.8	CRAN (R 4.2.3)
fastmap	1.1.1	CRAN (R 4.2.3)
forcats	1.0.0	CRAN (R 4.2.3)
foreign	0.8-84	CRAN (R 4.2.3)
fs	1.6.3	CRAN (R 4.2.3)
generics	0.1.3	CRAN (R 4.2.3)
ggplot2	3.4.4	CRAN (R 4.2.3)
glue	1.6.2	CRAN (R 4.2.3)
gtable	0.3.4	CRAN (R 4.2.3)
haven	2.5.3	CRAN (R 4.2.3)
Hmisc	5.1-0	CRAN (R 4.2.3)
hms	1.1.3	CRAN (R 4.2.3)
htmltools	0.5.7	CRAN (R 4.2.3)
htmlwidgets	1.6.2	CRAN (R 4.2.3)
httr	1.4.7	CRAN (R 4.2.3)
investr	1.4.2	CRAN (R 4.2.3)
jpeg	0.1-10	CRAN (R 4.2.2)
jquerylib	0.1.4	CRAN (R 4.2.3)
jsonlite	1.8.8	CRAN (R 4.2.3)
kableExtra	1.3.4	CRAN (R 4.2.3)
knitr	1.45	CRAN (R 4.2.3)
lattice	0.21-8	CRAN (R 4.2.3)
latticeExtra	0.6-30	CRAN (R 4.2.3)
lifecycle	1.0.4	CRAN (R 4.2.3)
lme4	1.1-35.1	CRAN (R 4.2.3)

package	version	source
lmerTest	3.1-3	CRAN (R 4.2.3)
lsr	0.5.2	CRAN (R 4.2.3)
ltm	1.2-0	CRAN (R 4.2.3)
lubridate	1.9.2	CRAN (R 4.2.3)
magrittr	2.0.3	CRAN (R 4.2.3)
MASS	7.3-60	CRAN (R 4.2.3)
matlib	0.9.6	CRAN (R 4.2.3)
Matrix	1.6-1	CRAN (R 4.2.3)
MCMCglmm	2.35	CRAN (R 4.2.3)
memoise	2.0.1	CRAN (R 4.2.3)
mgcv	1.9-0	CRAN (R 4.2.3)
minqa	1.2.6	CRAN (R 4.2.3)
modelr	0.1.11	CRAN (R 4.2.3)
munsell	0.5.0	CRAN (R 4.2.3)
nlme	3.1-163	CRAN (R 4.2.3)
nloptr	2.0.3	CRAN (R 4.2.3)
nlstools	2.0-0	CRAN (R 4.2.3)
nnet	7.3-19	CRAN (R 4.2.3)
numDeriv	2016.8-1.1	CRAN (R 4.2.0)
openxlsx	4.2.5.2	CRAN (R 4.2.3)
pbkrtest	0.5.2	CRAN (R 4.2.3)
pillar	1.9.0	CRAN (R 4.2.3)
pkgbuild	1.4.3	CRAN (R 4.2.3)
pkgconfig	2.0.3	CRAN (R 4.2.3)
pkgload	1.3.3	CRAN (R 4.2.3)
png	0.1-8	CRAN (R 4.2.2)
ppsr	0.0.2	CRAN (R 4.2.3)
prettyunits	1.2.0	CRAN (R 4.2.3)
processx	3.8.2	CRAN (R 4.2.3)
ps	1.7.5	CRAN (R 4.2.3)
pscl	1.5.5.1	CRAN (R 4.2.3)
purrr	1.0.2	CRAN (R 4.2.3)
R6	2.5.1	CRAN (R 4.2.3)
RColorBrewer	1.1-3	CRAN (R 4.2.0)
Rcpp	1.0.11	CRAN (R 4.2.3)
readr	2.1.4	CRAN (R 4.2.3)
readxl	1.4.3	CRAN (R 4.2.3)
remotes	2.4.2.1	CRAN (R 4.2.3)
reprex	2.0.2	CRAN (R 4.2.3)
rgl	1.2.1	CRAN (R 4.2.3)
rio	1.0.1	CRAN (R 4.2.3)
rlang	1.1.1	CRAN (R 4.2.3)
RLRsim	3.1-8	CRAN (R 4.2.3)
rmarkdown	2.25	CRAN (R 4.2.3)

package	version	source
rprojroot	2.0.4	CRAN (R 4.2.3)
rstudioapi	0.15.0	CRAN (R 4.2.3)
Rttf2pt1	1.3.12	CRAN (R 4.2.2)
rvest	1.0.3	CRAN (R 4.2.3)
sass	0.4.8	CRAN (R 4.2.3)
scales	1.3.0	CRAN (R 4.2.3)
sessioninfo	1.2.2	CRAN (R 4.2.3)
stringi	1.7.12	CRAN (R 4.2.2)
stringr	1.5.1	CRAN (R 4.2.3)
svglite	2.1.1	CRAN (R 4.2.3)
systemfonts	1.0.5	CRAN (R 4.2.3)
tensorA	0.36.2	CRAN (R 4.2.0)
testthat	3.1.10	CRAN (R 4.2.3)
tibble	3.2.1	CRAN (R 4.2.3)
tidyr	1.3.0	CRAN (R 4.2.3)
tidyselect	1.2.0	CRAN (R 4.2.3)
tidyverse	2.0.0	CRAN (R 4.2.3)
tzdb	0.4.0	CRAN (R 4.2.3)
usethis	2.2.2	CRAN (R 4.2.3)
utf8	1.2.3	CRAN (R 4.2.3)
vctrs	0.6.3	CRAN (R 4.2.3)
viridisLite	0.4.2	CRAN (R 4.2.3)
webshot	0.5.5	CRAN (R 4.2.3)
withr	2.5.2	CRAN (R 4.2.3)
xfun	0.39	CRAN (R 4.2.3)
xml2	1.3.6	CRAN (R 4.2.3)
xtable	1.8-4	CRAN (R 4.2.3)
yaml	2.3.7	CRAN (R 4.2.3)
zip	2.3.0	CRAN (R 4.2.3)

```
#> - Session info ---------------------------------------------------------------
#>  setting  value
#>  version  R version 4.2.3 (2023-03-15 ucrt)
#>  os       Windows 10 x64 (build 22631)
#>  system   x86_64, mingw32
#>  ui       RTerm
#>  language (EN)
#>  collate  English_United States.utf8
#>  ctype    English_United States.utf8
#>  tz       America/Los_Angeles
#>  date     2024-02-08
#>  pandoc   3.1.1 @ C:/Program Files/RStudio/resources/app/bin/quarto/bin/tools/ (via
#>
#> - Packages -------------------------------------------------------------------
```

```
#>  package     * version date (UTC) lib source
#>  bookdown      0.35    2023-08-09 [1] CRAN (R 4.2.3)
#>  cachem        1.0.8   2023-05-01 [1] CRAN (R 4.2.3)
#>  cli           3.6.1   2023-03-23 [1] CRAN (R 4.2.3)
#>  codetools     0.2-19  2023-02-01 [1] CRAN (R 4.2.3)
#>  colorspace    2.1-0   2023-01-23 [1] CRAN (R 4.2.3)
#>  desc          1.4.3   2023-12-10 [1] CRAN (R 4.2.3)
#>  devtools      2.4.5   2022-10-11 [1] CRAN (R 4.2.3)
#>  digest        0.6.31  2022-12-11 [1] CRAN (R 4.2.3)
#>  dplyr       * 1.1.2   2023-04-20 [1] CRAN (R 4.2.3)
#>  ellipsis      0.3.2   2021-04-29 [1] CRAN (R 4.2.3)
#>  evaluate      0.23    2023-11-01 [1] CRAN (R 4.2.3)
#>  fansi         1.0.4   2023-01-22 [1] CRAN (R 4.2.3)
#>  fastmap       1.1.1   2023-02-24 [1] CRAN (R 4.2.3)
#>  forcats     * 1.0.0   2023-01-29 [1] CRAN (R 4.2.3)
#>  fs            1.6.3   2023-07-20 [1] CRAN (R 4.2.3)
#>  generics      0.1.3   2022-07-05 [1] CRAN (R 4.2.3)
#>  ggplot2     * 3.4.4   2023-10-12 [1] CRAN (R 4.2.3)
#>  glue          1.6.2   2022-02-24 [1] CRAN (R 4.2.3)
#>  gtable        0.3.4   2023-08-21 [1] CRAN (R 4.2.3)
#>  hms           1.1.3   2023-03-21 [1] CRAN (R 4.2.3)
#>  htmltools     0.5.7   2023-11-03 [1] CRAN (R 4.2.3)
#>  htmlwidgets   1.6.2   2023-03-17 [1] CRAN (R 4.2.3)
#>  httpuv        1.6.11  2023-05-11 [1] CRAN (R 4.2.3)
#>  jpeg        * 0.1-10  2022-11-29 [1] CRAN (R 4.2.2)
#>  knitr         1.45    2023-10-30 [1] CRAN (R 4.2.3)
#>  later         1.3.1   2023-05-02 [1] CRAN (R 4.2.3)
#>  lifecycle     1.0.4   2023-11-07 [1] CRAN (R 4.2.3)
#>  lubridate   * 1.9.2   2023-02-10 [1] CRAN (R 4.2.3)
#>  magrittr      2.0.3   2022-03-30 [1] CRAN (R 4.2.3)
#>  memoise       2.0.1   2021-11-26 [1] CRAN (R 4.2.3)
#>  mime          0.12    2021-09-28 [1] CRAN (R 4.2.0)
#>  miniUI        0.1.1.1 2018-05-18 [1] CRAN (R 4.2.3)
#>  munsell       0.5.0   2018-06-12 [1] CRAN (R 4.2.3)
#>  pillar        1.9.0   2023-03-22 [1] CRAN (R 4.2.3)
#>  pkgbuild      1.4.3   2023-12-10 [1] CRAN (R 4.2.3)
#>  pkgconfig     2.0.3   2019-09-22 [1] CRAN (R 4.2.3)
#>  pkgload       1.3.3   2023-09-22 [1] CRAN (R 4.2.3)
#>  profvis       0.3.8   2023-05-02 [1] CRAN (R 4.2.3)
#>  promises      1.2.1   2023-08-10 [1] CRAN (R 4.2.3)
#>  purrr       * 1.0.2   2023-08-10 [1] CRAN (R 4.2.3)
#>  R6            2.5.1   2021-08-19 [1] CRAN (R 4.2.3)
#>  Rcpp          1.0.11  2023-07-06 [1] CRAN (R 4.2.3)
#>  readr       * 2.1.4   2023-02-10 [1] CRAN (R 4.2.3)
#>  remotes       2.4.2.1 2023-07-18 [1] CRAN (R 4.2.3)
#>  rlang         1.1.1   2023-04-28 [1] CRAN (R 4.2.3)
```

```
#>  rmarkdown     2.25    2023-09-18 [1] CRAN (R 4.2.3)
#>  rstudioapi    0.15.0  2023-07-07 [1] CRAN (R 4.2.3)
#>  scales      * 1.3.0   2023-11-28 [1] CRAN (R 4.2.3)
#>  sessioninfo   1.2.2   2021-12-06 [1] CRAN (R 4.2.3)
#>  shiny         1.7.5   2023-08-12 [1] CRAN (R 4.2.3)
#>  stringi       1.7.12  2023-01-11 [1] CRAN (R 4.2.2)
#>  stringr     * 1.5.1   2023-11-14 [1] CRAN (R 4.2.3)
#>  tibble      * 3.2.1   2023-03-20 [1] CRAN (R 4.2.3)
#>  tidyr       * 1.3.0   2023-01-24 [1] CRAN (R 4.2.3)
#>  tidyselect    1.2.0   2022-10-10 [1] CRAN (R 4.2.3)
#>  tidyverse   * 2.0.0   2023-02-22 [1] CRAN (R 4.2.3)
#>  timechange    0.2.0   2023-01-11 [1] CRAN (R 4.2.3)
#>  tzdb          0.4.0   2023-05-12 [1] CRAN (R 4.2.3)
#>  urlchecker    1.0.1   2021-11-30 [1] CRAN (R 4.2.3)
#>  usethis       2.2.2   2023-07-06 [1] CRAN (R 4.2.3)
#>  utf8          1.2.3   2023-01-31 [1] CRAN (R 4.2.3)
#>  vctrs         0.6.3   2023-06-14 [1] CRAN (R 4.2.3)
#>  withr         2.5.2   2023-10-30 [1] CRAN (R 4.2.3)
#>  xfun          0.39    2023-04-20 [1] CRAN (R 4.2.3)
#>  xtable        1.8-4   2019-04-21 [1] CRAN (R 4.2.3)
#>  yaml          2.3.7   2023-01-23 [1] CRAN (R 4.2.3)
#>
#>  [1] C:/Program Files/R/R-4.2.3/library
#>
#> ------------------------------------------------------------------------------
```

Chapter 2

Prerequisites

This chapter serves as a concise review of fundamental concepts in Matrix Theory and Probability Theory.

If you are confident in your understanding of these topics, you can proceed directly to the Descriptive Statistics section to begin exploring applied data analysis.

2.1 Matrix Theory

Matrix A represents the original matrix. It's a 2x2 matrix with elements a_{ij}, where i represents the row and j represents the column.

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

A' is the transpose of A. The transpose of a matrix flips its rows and columns.

$$A' = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{bmatrix}$$

Fundamental properties and rules of matrices, essential for understanding operations in linear algebra:

$$
\begin{aligned}
(\mathbf{ABC})' &= \mathbf{C}'\mathbf{B}'\mathbf{A}' && \text{(Transpose reverses order in a product)} \\
\mathbf{A}(\mathbf{B}+\mathbf{C}) &= \mathbf{AB}+\mathbf{AC} && \text{(Distributive property)} \\
\mathbf{AB} &\neq \mathbf{BA} && \text{(Multiplication is not commutative)} \\
(\mathbf{A}')' &= \mathbf{A} && \text{(Double transpose is the original matrix)} \\
(\mathbf{A}+\mathbf{B})' &= \mathbf{A}'+\mathbf{B}' && \text{(Transpose of a sum is the sum of transposes)} \\
(\mathbf{AB})' &= \mathbf{B}'\mathbf{A}' && \text{(Transpose reverses order in a product)} \\
(\mathbf{AB})^{-1} &= \mathbf{B}^{-1}\mathbf{A}^{-1} && \text{(Inverse reverses order in a product)} \\
\mathbf{A}+\mathbf{B} &= \mathbf{B}+\mathbf{A} && \text{(Addition is commutative)} \\
\mathbf{AA}^{-1} &= \mathbf{I} && \text{(Matrix times its inverse is identity)}
\end{aligned}
$$

These properties are critical in solving systems of equations, optimizing models, and performing data transformations.

If a matrix $\mathbf{A}$ has an inverse, it is called **invertible**. If $\mathbf{A}$ does not have an inverse, it is referred to as **singular**.

The product of two matrices $\mathbf{A}$ and $\mathbf{B}$ is computed as:

$$
\begin{aligned}
\mathbf{A} &= \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \\
&= \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & \sum_{i=1}^{3} a_{1i}b_{i2} & \sum_{i=1}^{3} a_{1i}b_{i3} \\ \sum_{i=1}^{3} a_{2i}b_{i1} & \sum_{i=1}^{3} a_{2i}b_{i2} & \sum_{i=1}^{3} a_{2i}b_{i3} \end{bmatrix}
\end{aligned}
$$

Quadratic Form

Let $\mathbf{a}$ be a 3×1 vector. The quadratic form involving a matrix $\mathbf{B}$ is given by:

$$
\mathbf{a}'\mathbf{Ba} = \sum_{i=1}^{3} \sum_{j=1}^{3} a_i b_{ij} a_j
$$

Length of a Vector

The **length** (or 2-norm) of a vector $\mathbf{a}$, denoted as $||\mathbf{a}||$, is defined as the square root of the inner product of the vector with itself:

$$
||\mathbf{a}|| = \sqrt{\mathbf{a}'\mathbf{a}}
$$

2.1.1 Rank of a Matrix

The **rank** of a matrix refers to:

- The dimension of the space spanned by its columns (or rows).

- The number of linearly independent columns or rows.

For an $n \times k$ matrix $\mathbf{A}$ and a $k \times k$ matrix $\mathbf{B}$, the following properties hold:

- $\text{rank}(\mathbf{A}) \leq \min(n, k)$
- $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{A}') = \text{rank}(\mathbf{A}'\mathbf{A}) = \text{rank}(\mathbf{A}\mathbf{A}')$
- $\text{rank}(\mathbf{AB}) = \min(\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B}))$
- $\mathbf{B}$ is invertible (non-singular) if and only if $\text{rank}(\mathbf{B}) = k$.

2.1.2 Inverse of a Matrix

In scalar algebra, if $a = 0$, then $1/a$ does not exist.

In matrix algebra, a matrix is invertible if it is **non-singular**, meaning it has a non-zero determinant and its inverse exists. A square matrix $\mathbf{A}$ is invertible if there exists another square matrix $\mathbf{B}$ such that:

$$\mathbf{AB} = \mathbf{I} \quad \text{(Identity Matrix)}.$$

In this case, $\mathbf{A}^{-1} = \mathbf{B}$.

For a 2×2 matrix:

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

The inverse is:

$$\mathbf{A}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

This inverse exists only if $ad - bc \neq 0$, where $ad - bc$ is the determinant of $\mathbf{A}$.

For a partitioned block matrix:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} (\mathbf{A} - \mathbf{BD^{-1}C})^{-1} & -(\mathbf{A} - \mathbf{BD^{-1}C})^{-1}\mathbf{BD^{-1}} \\ -\mathbf{D^{-1}C}(\mathbf{A} - \mathbf{BD^{-1}C})^{-1} & \mathbf{D^{-1}} + \mathbf{D^{-1}C}(\mathbf{A} - \mathbf{BD^{-1}C})^{-1}\mathbf{BD^{-1}} \end{bmatrix}$$

This formula assumes that $\mathbf{D}$ and $\mathbf{A} - \mathbf{BD^{-1}C}$ are invertible.

Properties of the Inverse for Non-Singular Matrices

1. $(\mathbf{A^{-1}})^{-1} = \mathbf{A}$

2. For a non-zero scalar b, $(\mathbf{bA})^{-1} = \mathbf{b^{-1}A^{-1}}$

3. For a matrix $\mathbf{B}$, $(\mathbf{BA})^{-\mathbf{1}} = \mathbf{B}^{-\mathbf{1}}\mathbf{A}^{-\mathbf{1}}$ (only if $\mathbf{B}$ is non-singular).

4. $(\mathbf{A}^{-\mathbf{1}})' = (\mathbf{A}')^{-\mathbf{1}}$ (the transpose of the inverse equals the inverse of the transpose).

5. Never notate $\mathbf{1}/\mathbf{A}$; use $\mathbf{A}^{-\mathbf{1}}$ instead.

Notes: - The determinant of a matrix determines whether it is invertible. For square matrices, a determinant of 0 means the matrix is **singular** and has no inverse.
- Always verify the conditions for invertibility, particularly when dealing with partitioned or block matrices.

2.1.3 Definiteness of a Matrix

A symmetric square $k \times k$ matrix $\mathbf{A}$ is classified based on the following conditions:

- **Positive Semi-Definite (PSD)**: $\mathbf{A}$ is PSD if, for any non-zero $k \times 1$ vector $\mathbf{x}$:
$$\mathbf{x}'\mathbf{A}\mathbf{x} \geq \mathbf{0}.$$

- **Negative Semi-Definite (NSD)**: $\mathbf{A}$ is NSD if, for any non-zero $k \times 1$ vector $\mathbf{x}$:
$$\mathbf{x}'\mathbf{A}\mathbf{x} \leq \mathbf{0}.$$

- **Indefinite**: $\mathbf{A}$ is indefinite if it is neither PSD nor NSD.

The **identity matrix** is always positive definite (PD).

Example

Let $\mathbf{x} = (x_1, x_2)'$, and consider a 2×2 identity matrix $\mathbf{I}$:

$$\begin{aligned}
\mathbf{x}'\mathbf{I}\mathbf{x} &= (x_1, x_2) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\
&= (x_1, x_2) \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\
&= x_1^2 + x_2^2 \geq 0.
\end{aligned}$$

Thus, $\mathbf{I}$ is PD because $\mathbf{x}'\mathbf{I}\mathbf{x} > 0$ for all non-zero $\mathbf{x}$.

Properties of Definiteness

1. Any variance-covariance matrix is PSD.
2. A matrix $\mathbf{A}$ is PSD if and only if there exists a matrix $\mathbf{B}$ such that:
$$\mathbf{A} = \mathbf{B}'\mathbf{B}.$$

3. If $\mathbf{A}$ is PSD, then $\mathbf{B}'\mathbf{A}\mathbf{B}$ is also PSD for any conformable matrix $\mathbf{B}$.

4. If $\mathbf{A}$ and $\mathbf{C}$ are non-singular, then $\mathbf{A} - \mathbf{C}$ is PSD if and only if $\mathbf{C}^{-1} - \mathbf{A}^{-1}$ is PSD.
5. If $\mathbf{A}$ is PD (or ND), then $\mathbf{A}^{-1}$ is also PD (or ND).

Notes

- An **indefinite** matrix $\mathbf{A}$ is neither PSD nor NSD. This concept does not have a direct counterpart in scalar algebra.
- If a square matrix is PSD and invertible, then it is PD.

Examples of Definiteness

1. **Invertible / Indefinite**:
$$\begin{bmatrix} -1 & 0 \\ 0 & 10 \end{bmatrix}$$

2. **Non-Invertible / Indefinite**:
$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

3. **Invertible / PSD**:
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

4. **Non-Invertible / PSD**:
$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

2.1.4 Matrix Calculus

Consider a scalar function $y = f(x_1, x_2, \dots, x_k) = f(x)$, where x is a $1 \times k$ row vector.

2.1.4.1 Gradient (First-Order Derivative)

The gradient, or the first-order derivative of $f(x)$ with respect to the vector x, is given by:

$$\frac{\partial f(x)}{\partial x} = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_k} \end{bmatrix}$$

2.1.4.2 Hessian (Second-Order Derivative)

The Hessian, or the second-order derivative of $f(x)$ with respect to x, is a symmetric matrix defined as:

$$\frac{\partial^2 f(x)}{\partial x \partial x'} = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_k} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_k \partial x_1} & \frac{\partial^2 f(x)}{\partial x_k \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_k^2} \end{bmatrix}$$

2.1.4.3 Derivative of a Scalar Function with Respect to a Matrix

Let $f(\mathbf{X})$ be a scalar function, where $\mathbf{X}$ is an $n \times p$ matrix. The derivative is:

$$\frac{\partial f(\mathbf{X})}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial f(\mathbf{X})}{\partial x_{11}} & \cdots & \frac{\partial f(\mathbf{X})}{\partial x_{1p}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(\mathbf{X})}{\partial x_{n1}} & \cdots & \frac{\partial f(\mathbf{X})}{\partial x_{np}} \end{bmatrix}$$

2.1.4.4 Common Matrix Derivatives

1. If $\mathbf{a}$ is a vector and $\mathbf{A}$ is a matrix independent of $\mathbf{y}$:
   - $\frac{\partial \mathbf{a}'\mathbf{y}}{\partial \mathbf{y}} = \mathbf{a}$
   - $\frac{\partial \mathbf{y}'\mathbf{y}}{\partial \mathbf{y}} = 2\mathbf{y}$
   - $\frac{\partial \mathbf{y}'\mathbf{A}\mathbf{y}}{\partial \mathbf{y}} = (\mathbf{A} + \mathbf{A}')\mathbf{y}$
2. If $\mathbf{X}$ is symmetric:
   - $\frac{\partial |\mathbf{X}|}{\partial x_{ij}} = \begin{cases} X_{ii}, & i = j \\ X_{ij}, & i \neq j \end{cases}$ where X_{ij} is the (i, j)-th cofactor of $\mathbf{X}$.
3. If $\mathbf{X}$ is symmetric and $\mathbf{A}$ is a matrix independent of $\mathbf{X}$:
   - $\frac{\partial \mathrm{tr}(\mathbf{X}\mathbf{A})}{\partial \mathbf{X}} = \mathbf{A} + \mathbf{A}' - \mathrm{diag}(\mathbf{A})$.
4. If $\mathbf{X}$ is symmetric, let $\mathbf{J}_{ij}$ be a matrix with 1 at the (i, j)-th position and 0 elsewhere:
   - $\frac{\partial \mathbf{X}^{-1}}{\partial x_{ij}} = \begin{cases} -\mathbf{X}^{-1}\mathbf{J}_{ii}\mathbf{X}^{-1}, & i = j \\ -\mathbf{X}^{-1}(\mathbf{J}_{ij} + \mathbf{J}_{ji})\mathbf{X}^{-1}, & i \neq j \end{cases}.$

2.1.5 Optimization in Scalar and Vector Spaces

Optimization is the process of finding the minimum or maximum of a function. The conditions for optimization differ depending on whether the function involves a scalar or a vector. Below is a comparison of scalar and vector optimization:

Condition	**Scalar Optimization**	**Vector Optimization**
First-Order Condition	$\frac{\partial f(x_0)}{\partial x} = 0$	$\frac{\partial f(x_0)}{\partial x} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$
Second-Order Condition	$\frac{\partial^2 f(x_0)}{\partial x^2} > 0$	$\frac{\partial^2 f(x_0)}{\partial x \partial x'} > 0$
For **convex** functions, this implies a **minimum**.		
For **concave** functions, this implies a **maximum**.	$\frac{\partial^2 f(x_0)}{\partial x^2} < 0$	$\frac{\partial^2 f(x_0)}{\partial x \partial x'} < 0$

Key Concepts

1. **First-Order Condition**:
   - The **first-order derivative** of the function must equal zero at a critical point. This holds for both scalar and vector functions:
     - In the scalar case, $\frac{\partial f(x)}{\partial x} = 0$ identifies critical points.
     - In the vector case, $\frac{\partial f(x)}{\partial x}$ is a **gradient vector**, and the condition is satisfied when all elements of the gradient are zero.
2. **Second-Order Condition**:
   - The **second-order derivative** determines whether the critical point is a minimum, maximum, or saddle point:
     - For scalar functions, $\frac{\partial^2 f(x)}{\partial x^2} > 0$ implies a **local minimum**, while $\frac{\partial^2 f(x)}{\partial x^2} < 0$ implies a **local maximum**.
     - For vector functions, the Hessian matrix $\frac{\partial^2 f(x)}{\partial x \partial x'}$ must be:
       * **Positive Definite**: For a **minimum** (convex function).
       * **Negative Definite**: For a **maximum** (concave function).
       * **Indefinite**: For a **saddle point** (neither minimum nor maximum).
3. **Convex and Concave Functions**:
   - A function $f(x)$ is:
     - **Convex** if $\frac{\partial^2 f(x)}{\partial x^2} > 0$ or the Hessian $\frac{\partial^2 f(x)}{\partial x \partial x'}$ is positive definite.
     - **Concave** if $\frac{\partial^2 f(x)}{\partial x^2} < 0$ or the Hessian is negative definite.
   - Convexity ensures global optimization for minimization problems, while concavity ensures global optimization for maximization problems.
4. **Hessian Matrix**:
   - In vector optimization, the Hessian $\frac{\partial^2 f(x)}{\partial x \partial x'}$ plays a crucial role in determining the nature of critical points:

  - Positive definite Hessian: All eigenvalues are positive.
  - Negative definite Hessian: All eigenvalues are negative.
  - Indefinite Hessian: Eigenvalues have mixed signs.

2.1.6 Cholesky Decomposition

In statistical analysis and numerical linear algebra, decomposing matrices into more tractable forms is crucial for efficient computation. One such important factorization is the **Cholesky Decomposition**. It applies to **Hermitian** (in the complex case) or **symmetric** (in the real case), **positive-definite** matrices.

Given an $n \times n$ positive-definite matrix A, the Cholesky Decomposition states:

$$A = LL^*,$$

where:

- L is a lower-triangular matrix with strictly positive diagonal entries.
- L^* denotes the *conjugate transpose* of L (simply the transpose L^T for real matrices).

Cholesky Decomposition is both computationally efficient and numerically stable, making it a go-to technique for many applications—particularly in statistics where we deal extensively with covariance matrices, linear systems, and probability distributions.

Before diving into how we compute a Cholesky Decomposition, we need to clarify what it means for a matrix to be *positive-definite*. For a real symmetric matrix A:

- A is *positive-definite* if for every nonzero vector x, we have

$$x^T A\, x > 0.$$

- Alternatively, you can characterize positive-definiteness by noting that all eigenvalues of A are strictly positive.

Many important matrices in statistics—particularly *covariance* or *precision* matrices—are both symmetric and positive-definite.

2.1.6.1 Existence

A real $n \times n$ matrix A that is symmetric and positive-definite always admits a Cholesky Decomposition $A = LL^T$. This theorem guarantees that for any covariance matrix in statistics—assuming it is valid (i.e., positive-definite)—we can decompose it via Cholesky.

2.1.6.2 Uniqueness

If we additionally require that the diagonal entries of L are *strictly positive*, then L is *unique*. That is, no other lower-triangular matrix with strictly positive diagonal entries will produce the same factorization. This uniqueness is helpful for ensuring consistent numerical outputs in software implementations.

2.1.6.3 Constructing the Cholesky Factor L

Given a real, symmetric, positive-definite matrix $A \in \mathbb{R}^{n\times n}$, we want to find the lower-triangular matrix L such that $A = LL^T$. One way to do this is by using a simple step-by-step procedure (often part of standard linear algebra libraries):

1. **Initialize** L to be an $n \times n$ zero matrix.
2. **Iterate** through the rows $i = 1, 2, \dots, n$:
   1. For each row i, compute
   $$L_{ii} = \sqrt{A_{ii} - \sum_{k=1}^{i-1} L_{ik}^2}.$$
   2. For each column $j = i+1, i+2, \dots, n$:
   $$L_{ji} = \frac{1}{L_{ii}}\Big(A_{ji} - \sum_{k=1}^{i-1} L_{jk}L_{ik}\Big).$$
   3. All other entries of L remain zero or are computed in subsequent steps.
3. **Result**: L is lower-triangular, and L^T is its transpose.

Cholesky Decomposition is roughly half the computational cost of a more general LU Decomposition. Specifically, it requires on the order of $\frac{1}{3}n^3$ floating-point operations (flops), making it significantly more efficient in practice than other decompositions for positive-definite systems.

2.1.6.4 Illustrative Example

Consider a small 3×3 positive-definite matrix:

$$A = \begin{pmatrix} 4 & 2 & 4 \\ 2 & 5 & 6 \\ 4 & 6 & 20 \end{pmatrix}.$$

We claim A is positive-definite (one could check by calculating principal minors or verifying $x^T Ax > 0$ for all $x \neq 0$). We find L step-by-step:

1. **Compute** L_{11}:

$$L_{11} = \sqrt{A_{11}} = \sqrt{4} = 2.$$

2. **Compute** L_{21} and L_{31}:
   - $L_{21} = \frac{A_{21}}{L_{11}} = \frac{2}{2} = 1.$
   - $L_{31} = \frac{A_{31}}{L_{11}} = \frac{4}{2} = 2.$
3. **Compute** L_{22}:

$$L_{22} = \sqrt{A_{22} - L_{21}^2} = \sqrt{5 - 1^2} = \sqrt{4} = 2.$$

4. **Compute** L_{32}:

$$L_{32} = \frac{A_{32} - L_{31}L_{21}}{L_{22}} = \frac{6 - (2)(1)}{2} = \frac{4}{2} = 2.$$

5. **Compute** L_{33}:

$$L_{33} = \sqrt{A_{33} - (L_{31}^2 + L_{32}^2)} = \sqrt{20 - (2^2 + 2^2)} = \sqrt{20 - 8} = \sqrt{12} = 2\sqrt{3}.$$

Thus,

$$L = \begin{pmatrix} 2 & 0 & 0 \\ 1 & 2 & 0 \\ 2 & 2 & 2\sqrt{3} \end{pmatrix}.$$

One can verify $LL^T = A$.

2.1.6.5 Applications in Statistics

2.1.6.5.1 Solving Linear Systems A common statistical problem is solving $Ax = b$ for x. For instance, in regression or in computing Bayesian posterior modes, we often need to solve linear equations with covariance or precision matrices. With $A = LL^T$:

1. **Forward Substitution**: Solve $L\,y = b$.
2. **Backward Substitution**: Solve $L^T x = y$.

This two-step process is more stable and efficient than directly inverting A (which is typically discouraged due to numerical issues).

2.1.6.5.2 Generating Correlated Random Vectors In simulation-based statistics (e.g., Monte Carlo methods), we often need to generate random draws from a **multivariate normal** distribution $\mathcal{N}(\mu, \Sigma)$, where Σ is the covariance matrix. The steps are:

1. Generate a vector $z \sim \mathcal{N}(0, I)$ of independent standard normal variables.
2. Compute $x = \mu + Lz$, where $\Sigma = LL^T$.

Then x has the desired covariance structure Σ. This technique is widely used in Bayesian statistics (e.g., MCMC sampling) and financial modeling (e.g., portfolio simulations).

2.1.6.5.3 Gaussian Processes and Kriging In Gaussian Process modeling (common in spatial statistics, machine learning, and geostatistics), we frequently work with large covariance matrices that describe the correlations between observed data points:

$$\Sigma = \begin{pmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n, x_1) & k(x_n, x_2) & \cdots & k(x_n, x_n) \end{pmatrix},$$

where $k(\cdot, \cdot)$ is a covariance (kernel) function. We may need to invert or factorize Σ repeatedly to evaluate the log-likelihood:

$$\log \mathcal{L}(\theta) \sim -\tfrac{1}{2}(y - m(\theta))^T \Sigma^{-1}(y - m(\theta)) - \tfrac{1}{2}\log|\Sigma|,$$

where $m(\theta)$ is the mean function and θ are parameters. Using the Cholesky factor L of Σ helps:

- Σ^{-1} can be implied by solving systems with L instead of explicitly computing the inverse.
- $\log|\Sigma|$ can be computed as $2\sum_{i=1}^{n} \log L_{ii}$.

Hence, Cholesky Decomposition becomes the backbone of Gaussian Process computations.

2.1.6.5.4 Bayesian Inference with Covariance Matrices Many Bayesian models—especially hierarchical models—assume a multivariate normal prior on parameters. Cholesky Decomposition is used to:

- Sample from these priors or from posterior distributions.
- Regularize large covariance matrices.
- Speed up Markov Chain Monte Carlo (MCMC) computations by factorizing covariance structures.

2.1.6.6 Other Notes

1. Numerical Stability Considerations

Cholesky Decomposition is considered more stable than a general LU Decomposition when applied to positive-definite matrices. Since no row or column pivots are required, rounding errors can be smaller. Of course, in practice, software implementations can vary, and extremely ill-conditioned matrices can still pose numerical challenges.

2. Why We *Don't* Usually Compute $\mathbf{A}^{-1}$

It is common in statistics (especially in older texts) to see formulas involving Σ^{-1}. However, computing an inverse explicitly is often discouraged because:

- It is numerically less stable.
- It requires more computations.
- Many tasks that *appear* to need Σ^{-1} can be done more efficiently by solving systems via the Cholesky factor L.

Hence, "**solve, don't invert**" is a common mantra. If you see an expression like $\Sigma^{-1}b$, you can use the Cholesky factors L and L^T to solve $\Sigma x = b$ by forward and backward substitution, bypassing the direct inverse calculation.

3. Further Variants and Extensions

- **Incomplete Cholesky**: Sometimes used in iterative solvers where a full Cholesky factorization is too expensive, especially for large sparse systems.
- **LDL^T Decomposition**: A variant that avoids taking square roots; used for positive *semi*-definite or indefinite systems, but with caution about pivoting strategies.

2.2 Probability Theory

2.2.1 Axioms and Theorems of Probability

1. Let S denote the sample space of an experiment. Then:

$$P[S] = 1$$

(The probability of the sample space is always 1.)

2. For any event A:

$$P[A] \geq 0$$

(Probabilities are always non-negative.)

3. Let $A_1, A_2, A_3, ...$ be a finite or infinite collection of mutually exclusive events. Then:

$$P[A_1 \cup A_2 \cup A_3 ...] = P[A_1] + P[A_2] + P[A_3] + ...$$

(The probability of the union of mutually exclusive events is the sum of their probabilities.)

4. The probability of the empty set is:

$$P[\emptyset] = 0$$

5. The complement rule:

$$P[A'] = 1 - P[A]$$

6. The probability of the union of two events:

$$P[A_1 \cup A_2] = P[A_1] + P[A_2] - P[A_1 \cap A_2]$$

2.2.1.1 Conditional Probability

The conditional probability of A given B is defined as:

$$P[A|B] = \frac{P[A \cap B]}{P[B]}, \quad \text{provided } P[B] \neq 0.$$

2.2.1.2 Independent Events

Two events A and B are independent if and only if:

1. $P[A \cap B] = P[A]P[B]$
2. $P[A|B] = P[A]$
3. $P[B|A] = P[B]$

A collection of events $A_1, A_2, \dots, A_n$ is independent if and only if every subcollection is independent.

2.2.1.3 Multiplication Rule

The probability of the intersection of two events can be calculated as:

$$P[A \cap B] = P[A|B]P[B] = P[B|A]P[A].$$

2.2.1.4 Bayes' Theorem

Let $A_1, A_2, \dots, A_n$ be a collection of mutually exclusive events whose union is S, and let B be an event with $P[B] \neq 0$. Then, for any event A_j $(j = 1, 2, \dots, n)$:

$$P[A_j|B] = \frac{P[B|A_j]P[A_j]}{\sum_{i=1}^{n} P[B|A_i]P[A_i]}.$$

2.2.1.5 Jensen's Inequality

- If $g(x)$ is convex, then:

$$E[g(X)] \geq g(E[X])$$

- If $g(x)$ is concave, then:

$$E[g(X)] \leq g(E[X]).$$

Jensen's inequality provides a useful way to demonstrate why the standard error calculated using the sample standard deviation (s) as a proxy for the population standard deviation (σ) is a biased estimator.

- The population standard deviation σ is defined as:

$$\sigma = \sqrt{\mathbb{E}[(X-\mu)^2]},$$

  where $\mu = \mathbb{E}[X]$ is the population mean.

- The sample standard deviation s is given by:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2},$$

  where $\bar{X}$ is the sample mean.

- When s is used as an estimator for σ, the expectation involves the square root function, which is concave.

Applying Jensen's Inequality

The standard error formula involves the square root:

$$\sqrt{\mathbb{E}[s^2]}.$$

However, because the square root function is concave, Jensen's inequality implies:

$$\sqrt{\mathbb{E}[s^2]} \leq \mathbb{E}[\sqrt{s^2}] = \mathbb{E}[s].$$

This inequality shows that the expected value of s (the sample standard deviation) systematically underestimates the population standard deviation σ.

Quantifying the Bias

The bias arises because:

$$\mathbb{E}[s] \neq \sigma.$$

To correct this bias, we note that the sample standard deviation is related to the population standard deviation by:

$$\mathbb{E}[s] = \sigma \cdot \sqrt{\frac{n-1}{n}},$$

where n is the sample size. This bias decreases as n increases, and the estimator becomes asymptotically unbiased.

By leveraging Jensen's inequality, we observe that the concavity of the square root function ensures that s is a biased estimator of σ, systematically underestimating the population standard deviation.

2.2.1.6 Law of Iterated Expectation

The **Law of Iterated Expectation** states that for random variables X and Y:

$$E(X) = E(E(X|Y)).$$

This means the expected value of X can be obtained by first calculating the conditional expectation $E(X|Y)$ and then taking the expectation of this quantity over the distribution of Y.

2.2.1.7 Correlation and Independence

The strength of the relationship between random variables can be ranked from strongest to weakest as:

1. **Independence**:
   - $f(x, y) = f_X(x)f_Y(y)$
   - $f_{Y|X}(y|x) = f_Y(y)$ and $f_{X|Y}(x|y) = f_X(x)$
   - $E[g_1(X)g_2(Y)] = E[g_1(X)]E[g_2(Y)]$
2. **Mean Independence** (implied by independence):
   - Y is mean independent of X if:

     $$E[Y|X] = E[Y].$$

   - $E[Xg(Y)] = E[X]E[g(Y)]$
3. **Uncorrelatedness** (implied by independence and mean independence):
   - $\text{Cov}(X, Y) = 0$
   - $\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y)$
   - $E[XY] = E[X]E[Y]$

2.2.2 Central Limit Theorem

The **Central Limit Theorem** states that for a sufficiently large sample size ($n \geq 25$), the sampling distribution of the sample mean or proportion approaches a normal distribution, regardless of the population's original distribution.

Let $X_1, X_2, \ldots, X_n$ be a random sample of size n from a distribution X with mean μ and variance σ^2. Then, for large n:

1. The sample mean $\bar{X}$ is approximately normal:

$$\mu_{\bar{X}} = \mu, \quad \sigma^2_{\bar{X}} = \frac{\sigma^2}{n}.$$

2. The sample proportion $\hat{p}$ is approximately normal:

$$\mu_{\hat{p}} = p, \quad \sigma^2_{\hat{p}} = \frac{p(1-p)}{n}.$$

3. The difference in sample proportions $\hat{p}_1 - \hat{p}_2$ is approximately normal:

$$\mu_{\hat{p}_1 - \hat{p}_2} = p_1 - p_2, \quad \sigma^2_{\hat{p}_1 - \hat{p}_2} = \frac{p_1(1-p_1)}{n_1} + \frac{p_2(1-p_2)}{n_2}.$$

4. The difference in sample means $\bar{X}_1 - \bar{X}_2$ is approximately normal:

$$\mu_{\bar{X}_1 - \bar{X}_2} = \mu_1 - \mu_2, \quad \sigma^2_{\bar{X}_1 - \bar{X}_2} = \frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}.$$

5. The following random variables are approximately standard normal:
   - $\frac{\bar{X}-\mu}{\sigma/\sqrt{n}}$
   - $\frac{\hat{p}-p}{\sqrt{\frac{p(1-p)}{n}}}$
   - $\frac{(\hat{p}_1-\hat{p}_2)-(p_1-p_2)}{\sqrt{\frac{p_1(1-p_1)}{n_1}+\frac{p_2(1-p_2)}{n_2}}}$
   - $\frac{(\bar{X}_1-\bar{X}_2)-(\mu_1-\mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}}}$

2.2.2.1 Limiting Distribution of the Sample Mean

If $\{X_i\}_{i=1}^n$ is an iid random sample from a distribution with finite mean μ and finite variance σ^2, the sample mean $\bar{X}$ scaled by $\sqrt{n}$ has the following limiting distribution:

$$\sqrt{n}(\bar{X} - \mu) \xrightarrow{d} N(0, \sigma^2).$$

Standardizing the sample mean gives:

$$\frac{\sqrt{n}(\bar{X} - \mu)}{\sigma} \xrightarrow{d} N(0, 1).$$

Notes:

- The CLT holds for most random samples from any distribution (continuous, discrete, or unknown).
- It extends to the multivariate case: A random sample of a random vector converges to a multivariate normal distribution.

2.2.2.2 Asymptotic Variance and Limiting Variance

Asymptotic Variance (Avar):

$$Avar(\sqrt{n}(\bar{X} - \mu)) = \sigma^2.$$

- Refers to the variance of the limiting distribution of an estimator as the sample size (n) approaches infinity.
- It characterizes the variability of the scaled estimator $\sqrt{n}(\bar{x} - \mu)$ in its asymptotic distribution (e.g., normal distribution).

Limiting Variance ($\lim_{n\to\infty} Var$)

$$\lim_{n\to\infty} Var(\sqrt{n}(\bar{x} - \mu)) = \sigma^2$$

- Represents the value that the actual variance of $\sqrt{n}(\bar{x} - \mu)$ converges to as $n \to \infty$.

For a well-behaved estimator,

$$Avar(\sqrt{n}(\bar{X} - \mu)) = \lim_{n\to\infty} Var(\sqrt{n}(\bar{x} - \mu)) = \sigma^2.$$

However, **asymptotic variance is not necessarily equal to the limiting value of the variance** because asymptotic variance is derived from the limiting distribution, while limiting variance is a convergence result of the sequence of variances.

$$Avar(.) \neq lim_{n\to\infty} Var(.)$$

- Both the asymptotic variance $Avar$ and the limiting variance $\lim_{n\to\infty} Var$ are numerically equal to σ^2, but their conceptual definitions differ.
- $Avar(\cdot) \neq \lim_{n\to\infty} Var(\cdot)$. This emphasizes that while the numerical result may match, their derivation and meaning differ:
  - $Avar$ depends on the asymptotic (large-sample) distribution of the estimator.
  - $\lim_{n\to\infty} Var(\cdot)$ involves the sequence of variances as n grows.

Cases where the two do not match:

1. **Sample Quantiles**: Consider the sample quantile of order p, for some $0 < p < 1$. Under regularity conditions, the asymptotic distribution of the sample quantile is normal, with a variance that depends on p and the density of the distribution at the p-th quantile. However, the variance of the sample quantile itself does not necessarily converge to this limit as the sample size grows.
2. **Bootstrap Methods**: When using bootstrapping techniques to estimate the distribution of a statistic, the bootstrap distribution might converge to a different limiting distribution than the original statistic. In these cases, the variance of the bootstrap distribution (or the bootstrap variance) might differ from the limiting variance of the original statistic.
3. **Statistics with Randomly Varying Asymptotic Behavior**: In some cases, the asymptotic behavior of a statistic can vary randomly depending on the sample path. For such statistics, the asymptotic variance might not provide a consistent estimate of the limiting variance.
4. **M-estimators with Varying Asymptotic Behavior**: M-estimators can sometimes have different asymptotic behaviors depending on the tail behavior of the underlying distribution. For heavy-tailed distributions, the variance of the estimator might not stabilize even as the sample size grows large, making the asymptotic variance different from the variance of any limiting distribution.

2.2.3 Random Variable

Random variables can be categorized as either **discrete** or **continuous**, with distinct properties and functions defining each type.

	Discrete Variable	**Continuous Variable**
Definition	A random variable is discrete if it can assume at most a finite or countably infinite number of values.	A random variable is continuous if it can assume any value in some interval or intervals of real numbers, with $P(X = x) = 0$.
Density Function	A function f is called a density for X if: 1. $f(x) \geq 0$ 2. $\sum_x f(x) = 1$ 3. $f(x) = P(X = x)$ for x real	A function f is called a density for X if: 1. $f(x) \geq 0$ for x real 2. $\int_{-\infty}^{\infty} f(x)\,dx = 1$ 3. $P[a \leq X \leq b] = \int_a^b f(x)\,dx$ for a, b real
Cumulative Distribution Function	$F(x) = P(X \leq x)$	$F(x) = P(X \leq x) = \int_{-\infty}^{x} f(t)\,dt$
$E[H(X)]$	$\sum_x H(x)f(x)$	$\int_{-\infty}^{\infty} H(x)f(x)\,dx$
$\mu = E[X]$	$\sum_x xf(x)$	$\int_{-\infty}^{\infty} xf(x)\,dx$

	Discrete Variable	**Continuous Variable**
Ordinary Moments	$\sum_x x^k f(x)$	$\int_{-\infty}^{\infty} x^k f(x)\,dx$
Moment Generating Function	$m_X(t) = E[e^{tX}] = \sum_x e^{tx} f(x)$	$m_X(t) = E[e^{tX}] = \int_{-\infty}^{\infty} e^{tx} f(x)\,dx$

Expected Value Properties

- $E[c] = c$ for any constant c.
- $E[cX] = cE[X]$ for any constant c.
- $E[X+Y] = E[X] + E[Y]$.
- $E[XY] = E[X]E[Y]$ (if X and Y are independent).

Variance Properties

- $\mathrm{Var}(c) = 0$ for any constant c.
- $\mathrm{Var}(cX) = c^2\mathrm{Var}(X)$ for any constant c.
- $\mathrm{Var}(X) \geq 0$.
- $\mathrm{Var}(X) = E[X^2] - (E[X])^2$.
- $\mathrm{Var}(X+c) = \mathrm{Var}(X)$.
- $\mathrm{Var}(X+Y) = \mathrm{Var}(X) + \mathrm{Var}(Y)$ (if X and Y are independent).

The standard deviation σ is given by:

$$\sigma = \sqrt{\sigma^2} = \sqrt{\mathrm{Var}(X)}.$$

2.2.3.1 Multivariate Random Variables

Suppose $y_1, \dots, y_p$ are random variables with means $\mu_1, \dots, \mu_p$. Then:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_p \end{bmatrix}, \quad E[\mathbf{y}] = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_p \end{bmatrix} = \mu.$$

The covariance between y_i and y_j is $\sigma_{ij} = \mathrm{Cov}(y_i, y_j)$. The variance-covariance (or dispersion) matrix is:

$$= (\sigma_{ij}) = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \cdots & \sigma_{pp} \end{bmatrix}.$$

And is symmetric with $(p+1)p/2$ unique parameters.

Alternatively, let $u_{p\times 1}$ and $v_{v\times 1}$ be random vectors with means $_{\mathbf{u}}$ and $_{\mathbf{v}}$. then

$$_{\mathbf{uv}} = cov(\mathbf{u}, \mathbf{v}) = E[(\mathbf{u} - \ _{\mathbf{u}})(\mathbf{v} - \ _{\mathbf{v}})']$$

$\Sigma_{uv} \neq \Sigma_{vu}$ (but $\Sigma_{uv} = \Sigma'_{vu}$)

Properties of Covariance Matrices

1. **Symmetry**: $' = \ .$
2. **Eigen-Decomposition** (spectral decomposition,symmetric decomposition): $= \ ,$ where is a matrix of eigenvectors such that $' = \mathbf{I}$ (orthonormal), and is a diagonal matrix with eigenvalues $(\lambda_1, ..., \lambda_p)$ on the diagonal.
3. **Non-Negative Definiteness**: $\mathbf{a}\ \mathbf{a} \geq 0$ for any $\mathbf{a} \in R^p$. Equivalently, the eigenvalues of , $\lambda_1 \geq ... \geq \lambda_p \geq 0$
4. **Generalized Variance**: $| \ | = \lambda_1 ... \lambda_p \geq 0$.
5. **Trace**: $\text{tr}(\ ) = \lambda_1 + \cdots + \lambda_p = \sigma_{11} + \cdots + \sigma_{pp} = \sum \sigma_{ii} =$ sum of variances (total variance).

Note: is required to be positive definite. This implies that all eigenvalues are positive, and has an inverse $^{-1}$, such that $^{-1} = \mathbf{I}_{p\times p} = \ ^{-1}$

2.2.3.2 Correlation Matrices

The correlation coefficient ρ_{ij} and correlation matrix $\mathbf{R}$ are defined as:

$$\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}, \quad \mathbf{R} = \begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1p} \\ \rho_{21} & 1 & \cdots & \rho_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{p1} & \rho_{p2} & \cdots & 1 \end{bmatrix}.$$

where $\rho_{ii} = 1 \forall i$

2.2.3.3 Linear Transformations

Let $\mathbf{A}$ and $\mathbf{B}$ be matrices of constants, and $\mathbf{c}$ and $\mathbf{d}$ be vectors of constants. Then:

- $E[\mathbf{Ay} + \mathbf{c}] = \mathbf{A}\ _{\mathbf{y}} + \mathbf{c}$.
- $\text{Var}(\mathbf{Ay} + \mathbf{c}) = \mathbf{A}\ _{\mathbf{y}}\mathbf{A}'$.
- $\text{Cov}(\mathbf{Ay} + \mathbf{c}, \mathbf{By} + \mathbf{d}) = \mathbf{A}\ _{\mathbf{y}}\mathbf{B}'$.

2.2.4 Moment Generating Function

2.2.4.1 Properties of the Moment Generating Function

1. $\left.\frac{d^k(m_X(t))}{dt^k}\right|_{t=0} = E[X^k]$ (The k-th derivative at $t = 0$ gives the k-th moment of X).
2. $\mu = E[X] = m'_X(0)$ (The first derivative at $t = 0$ gives the mean).
3. $E[X^2] = m''_X(0)$ (The second derivative at $t = 0$ gives the second moment).

2.2.4.2 Theorems Involving MGFs

Let $X_1, X_2, \dots, X_n, Y$ be random variables with MGFs $m_{X_1}(t), m_{X_2}(t), \dots, m_{X_n}(t), m_Y(t)$:

1. If $m_{X_1}(t) = m_{X_2}(t)$ for all t in some open interval about 0, then X_1 and X_2 have the same distribution.
2. If $Y = \alpha + \beta X_1$, then:
$$m_Y(t) = e^{\alpha t} m_{X_1}(\beta t).$$
3. If $X_1, X_2, \dots, X_n$ are independent and $Y = \alpha_0 + \alpha_1 X_1 + \alpha_2 X_2 + \cdots + \alpha_n X_n$, where $\alpha_0, \alpha_1, \dots, \alpha_n$ are constants, then:
$$m_Y(t) = e^{\alpha_0 t} m_{X_1}(\alpha_1 t) m_{X_2}(\alpha_2 t) \dots m_{X_n}(\alpha_n t).$$
4. Suppose $X_1, X_2, \dots, X_n$ are independent normal random variables with means $\mu_1, \mu_2, \dots, \mu_n$ and variances $\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2$. If $Y = \alpha_0 + \alpha_1 X_1 + \alpha_2 X_2 + \cdots + \alpha_n X_n$, then:
   - Y is normally distributed.
   - Mean: $\mu_Y = \alpha_0 + \alpha_1 \mu_1 + \alpha_2 \mu_2 + \cdots + \alpha_n \mu_n$.
   - Variance: $\sigma_Y^2 = \alpha_1^2 \sigma_1^2 + \alpha_2^2 \sigma_2^2 + \cdots + \alpha_n^2 \sigma_n^2$.

2.2.5 Moments

Moment	Uncentered	Centered
1st	$E[X] = \mu = \text{Mean}(X)$	
2nd	$E[X^2]$	$E[(X-\mu)^2] = \text{Var}(X) = \sigma^2$
3rd	$E[X^3]$	$E[(X-\mu)^3]$
4th	$E[X^4]$	$E[(X-\mu)^4]$

- **Skewness**: $\text{Skewness}(X) = \frac{E[(X-\mu)^3]}{\sigma^3}$
  - **Definition**: Skewness measures the asymmetry of a probability distribution around its mean.
  - **Interpretation**:

    * **Positive skewness**: The right tail (higher values) is longer or heavier than the left tail.
    * **Negative skewness**: The left tail (lower values) is longer or heavier than the right tail.
    * **Zero skewness**: The data is symmetric.

- **Kurtosis**: $\text{Kurtosis}(X) = \frac{E[(X-\mu)^4]}{\sigma^4}$
  - **Definition**: Kurtosis measures the "tailedness" or the heaviness of the tails of a probability distribution.
  - Excess kurtosis (often reported) is the kurtosis minus 3 (to compare against the normal distribution's kurtosis of 3).
  - **Interpretation**:
    * **High kurtosis (>3)**: Heavy tails, more extreme outliers.
    * **Low kurtosis (<3)**: Light tails, fewer outliers.
    * **Normal distribution kurtosis = 3**: Benchmark for comparison.

2.2.6 Skewness

Skewness measures the asymmetry of the distribution:

1. **Positive skew**: The right side (high values) is stretched out.
   - Positive skew occurs when the right tail (higher values) of the distribution is longer or heavier.
   - **Examples**:
     - **Income Distribution**: In many countries, most people earn a moderate income, but a small fraction of ultra-high earners stretches the distribution's right tail.
     - **Housing Prices**: Most homes may be around an affordable price, but a few extravagant mansions create a very long (and expensive) upper tail.

```r
# Load required libraries
library(ggplot2)

# Simulate data for positive skew
set.seed(123)
positive_skew_income <-
    rbeta(1000, 5, 2) * 100  # Income distribution example
positive_skew_housing <-
    rbeta(1000, 5, 2) * 1000  # Housing prices example
```

```r
# Combine data
data_positive_skew <- data.frame(
    value = c(positive_skew_income, positive_skew_housing),
    example = c(rep("Income Distribution", 1000), rep("Housing Prices", 1000))
)

# Plot positive skew
ggplot(data_positive_skew, aes(x = value, fill = example)) +
    geom_histogram(bins = 30,
                   alpha = 0.7,
                   position = "identity") +
    facet_wrap( ~ example, scales = "free") +
    labs(title = "Visualization of Positive Skew",
         x = "Value",
         y = "Frequency") +
    theme_minimal()
```

- In the **Income Distribution** example, most people earn moderate incomes, but a few high earners stretch the right tail.
- In the **Housing Prices** example, most homes are reasonably priced, but a few mansions create a long, expensive right tail.

2. Negative Skew (Left Skew)

- Negative skew occurs when the left tail (lower values) of the distribution is longer or heavier.

- **Examples**:

- **Scores on an Easy Test**: If an exam is very easy, most students score quite high, and only a few students score low, creating a left tail.

- **Age of Retirement**: Most people might retire around a common age (say 65+), with fewer retiring very early (stretching the left tail).

```r
# Simulate data for negative skew
set.seed(123)
negative_skew_test <-
    10 - rbeta(1000, 5, 2) * 10  # Easy test scores example
negative_skew_retirement <-
    80 - rbeta(1000, 5, 2) * 20  # Retirement age example

# Combine data
data_negative_skew <- data.frame(
    value = c(negative_skew_test, negative_skew_retirement),
    example = c(rep("Easy Test Scores", 1000), rep("Retirement Age", 1000))
)

# Plot negative skew
ggplot(data_negative_skew, aes(x = value, fill = example)) +
    geom_histogram(bins = 30,
                   alpha = 0.7,
                   position = "identity") +
    facet_wrap( ~ example, scales = "free") +
    labs(title = "Visualization of Negative Skew",
         x = "Value",
         y = "Frequency") +
    theme_minimal()
```

- In the **Easy Test Scores** example, most students perform well, but a few low scores stretch the left tail.
- In the **Retirement Age** example, most people retire around the same age, but a small number of individuals retire very early, stretching the left tail.

2.2.7 Kurtosis

Kurtosis measures the "peakedness" or heaviness of the tails:

- **High kurtosis**: Tall, sharp peak with heavy tails.
  - **Example**: Financial market returns during a crisis (extreme losses or gains).
- **Low kurtosis**: Flatter peak with thinner tails.
  - **Example**: Human height distribution (fewer extreme deviations from the mean).

```r
# Simulate data for kurtosis
low_kurtosis <- runif(1000, 0, 10)  # Low kurtosis
high_kurtosis <- c(rnorm(900, 5, 1), rnorm(100, 5, 5))  # High kurtosis

# Combine data
data_kurtosis <- data.frame(
  value = c(low_kurtosis, high_kurtosis),
  kurtosis_type = c(rep("Low Kurtosis (Height Distribution)", 1000),
                    rep("High Kurtosis (Market Returns)", 1000))
```

```
)

# Plot kurtosis
ggplot(data_kurtosis, aes(x = value, fill = kurtosis_type)) +
  geom_histogram(bins = 30, alpha = 0.7, position = "identity") +
  facet_wrap(~kurtosis_type) +
  labs(
    title = "Visualization of Kurtosis",
    x = "Value",
    y = "Frequency"
  ) +
  theme_minimal()
```

- The left panel shows **low kurtosis**, similar to the distribution of human height, which has a flatter peak and thinner tails.
- The right panel shows **high kurtosis**, reflecting financial market returns, where there are more extreme outliers in gains or losses.

2.2.7.1 Conditional Moments

For a random variable Y given $X = x$:

1. **Expected Value**:

$$E[Y|X = x] = \begin{cases} \sum_y y f_Y(y|x) & \text{for discrete RV,} \\ \int_y y f_Y(y|x)\, dy & \text{for continuous RV.} \end{cases}$$

2. **Variance**:

$$\mathrm{Var}(Y|X=x) = \begin{cases} \sum_y (y - E[Y|X=x])^2 f_Y(y|x) & \text{for discrete RV,} \\ \int_y (y - E[Y|X=x])^2 f_Y(y|x)\, dy & \text{for continuous RV.} \end{cases}$$

2.2.7.2 Multivariate Moments

1. **Expected Value**:

$$E\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} E[X] \\ E[Y] \end{bmatrix} = \begin{bmatrix} \mu_X \\ \mu_Y \end{bmatrix}$$

2. **Variance-Covariance Matrix**:

$$\begin{aligned} \mathrm{Var}\begin{bmatrix} X \\ Y \end{bmatrix} &= \begin{bmatrix} \mathrm{Var}(X) & \mathrm{Cov}(X,Y) \\ \mathrm{Cov}(X,Y) & \mathrm{Var}(Y) \end{bmatrix} \\ &= \begin{bmatrix} E[(X-\mu_X)^2] & E[(X-\mu_X)(Y-\mu_Y)] \\ E[(X-\mu_X)(Y-\mu_Y)] & E[(Y-\mu_Y)^2] \end{bmatrix} \end{aligned}$$

2.2.7.3 Properties of Moments

1. $E[aX + bY + c] = aE[X] + bE[Y] + c$
2. $\mathrm{Var}(aX + bY + c) = a^2\mathrm{Var}(X) + b^2\mathrm{Var}(Y) + 2ab\mathrm{Cov}(X,Y)$
3. $\mathrm{Cov}(aX + bY, cX + dY) = ac\mathrm{Var}(X) + bd\mathrm{Var}(Y) + (ad + bc)\mathrm{Cov}(X,Y)$
4. Correlation: $\rho_{XY} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y}$

2.2.8 Distributions

2.2.8.1 Conditional Distributions

$$f_{X|Y}(x|y) = \frac{f(x,y)}{f_Y(y)}$$

If X and Y are independent:

$$f_{X|Y}(x|y) = f_X(x).$$

2.2.8.2 Discrete Distributions

2.2.8.2.1 Bernoulli Distribution A random variable X follows a Bernoulli distribution, denoted as $X \sim \mathrm{Bernoulli}(p)$, if it represents a single trial with:

- Success probability p
- Failure probability $q = 1 - p$.

Density Function

$$f(x) = p^x(1-p)^{1-x}, \quad x \in \{0, 1\}$$

CDF: Use table or manual computation.

PDF

```
hist(
    mc2d::rbern(1000, prob = 0.5),
    main = "Histogram of Bernoulli Distribution",
    xlab = "Value",
    ylab = "Frequency"
)
```

Mean

$$\mu = E[X] = p$$

Variance

$$\sigma^2 = \mathrm{Var}(X) = p(1-p)$$

2.2.8.2.2 Binomial Distribution $X \sim B(n, p)$ is the number of successes in n independent Bernoulli trials, where:

- n is the number of trials

- p is the success probability.
- The trials are identical and independent, and probability of success (p) and probability of failure ($q = 1 - p$) remains the same for all trials.

Density Function

$$f(x) = \binom{n}{x} p^x (1-p)^{n-x}, \quad x = 0, 1, \ldots, n$$

PDF

```
hist(
    rbinom(1000, size = 100, prob = 0.5),
    main = "Histogram of Binomial Distribution",
    xlab = "Value",
    ylab = "Frequency"
)
```

MGF

$$m_X(t) = (1 - p + pe^t)^n$$

Mean

$$\mu = np$$

Variance

$$\sigma^2 = np(1-p)$$

2.2.8.2.3 Poisson Distribution $X \sim \text{Poisson}(\lambda)$ models the number of occurrences of an event in a fixed interval, with average rate λ.

- Arises with Poisson process, which involves observing discrete events in a continuous "interval" of time, length, or space.
- The random variable X is the number of occurrences of the event within an interval of s units.
- The parameter λ is the average number of occurrences of the event in question per measurement unit. For the distribution, we use the parameter $k = \lambda s$.

Density Function

$$f(x) = \frac{e^{-k}k^x}{x!}, \quad x = 0, 1, 2, ...$$

CDF

PDF

```
hist(rpois(1000, lambda = 5),
     main = "Histogram of Poisson Distribution",
     xlab = "Value",
     ylab = "Frequency")
```

MGF

$$m_X(t) = e^{k(e^t-1)}$$

Mean

$$\mu = E(X) = k$$

Variance

$$\sigma^2 = Var(X) = k$$

2.2.8.2.4 Geometric Distribution $X \sim \mathrm{G}(p)$ models the number of trials needed to obtain the first success, with:

- p: probability of success
- $q = 1 - p$: probability of failure.
- The experiment consists of a series of trails. The outcome of each trial can be classed as being either a "success" (s) or "failure" (f). (i.e., Bernoulli trial).
- The trials are identical and independent in the sense that the outcome of one trial has no effect on the outcome of any other (i..e, lack of memory - momerylessness). The probability of success (p) and probability of failure ($q = 1 - p$) remains the same from trial to trial.

Density Function

$$f(x) = p(1-p)^{x-1}, \quad x = 1, 2, ...$$

CDF

$$F(x) = 1 - (1-p)^x$$

PDF

```
hist(rgeom(1000, prob = 0.5),
     main = "Histogram of Geometric Distribution",
     xlab = "Value",
     ylab = "Frequency")
```

MGF

$$m_X(t) = \frac{pe^t}{1-(1-p)e^t}, \quad t < -\ln(1-p)$$

Mean

$$\mu = \frac{1}{p}$$

Variance

$$\sigma^2 = \frac{1-p}{p^2}$$

2.2.8.2.5 Hypergeometric Distribution $X \sim \mathrm{H}(N, r, n)$ models the number of successes in a sample of size n drawn without replacement from a population of size N, where:

- r objects have the trait of interest
- $N - r$ do not have the trait.

Density Function

$$f(x) = \frac{\binom{r}{x}\binom{N-r}{n-x}}{\binom{N}{n}}, \quad \max(0, n-(N-r)) \leq x \leq \min(n, r)$$

PDF

```
hist(
    rhyper(1000, m = 50, n = 20, k = 30),
    main = "Histogram of Hypergeometric Distribution",
    xlab = "Value",
    ylab = "Frequency"
)
```

Mean

$$\mu = E[X] = \frac{nr}{N}$$

Variance

$$\sigma^2 = \mathrm{Var}(X) = n\frac{r}{N}\frac{N-r}{N}\frac{N-n}{N-1}$$

Note: For large N (when $\frac{n}{N} \leq 0.05$), the hypergeometric distribution can be approximated by a binomial distribution with $p = \frac{r}{N}$.

2.2.8.3 Continuous Distributions

2.2.8.3.1 Uniform Distribution Defined over an interval (a, b), where the probabilities are "equally likely" for subintervals of equal length.

Density Function:

$$f(x) = \frac{1}{b-a}, \quad a < x < b$$

CDF

$$F(x) = \begin{cases} 0 & \text{if } x < a \\ \frac{x-a}{b-a} & a \leq x \leq b \\ 1 & \text{if } x > b \end{cases}$$

PDF

```r
hist(
    runif(1000, min = 0, max = 1),
    main = "Histogram of Uniform Distribution",
    xlab = "Value",
    ylab = "Frequency"
)
```

MGF

$$m_X(t) = \begin{cases} \frac{e^{tb}-e^{ta}}{t(b-a)} & \text{if } t \neq 0 \\ 1 & \text{if } t = 0 \end{cases}$$

Mean

$$\mu = E[X] = \frac{a+b}{2}$$

Variance

$$\sigma^2 = \text{Var}(X) = \frac{(b-a)^2}{12}$$

2.2.8.3.2 Gamma Distribution The gamma distribution is used to define the exponential and χ^2 distributions.

The **gamma function** is defined as:

$$\Gamma(\alpha) = \int_0^\infty z^{\alpha-1} e^{-z} dz, \quad \alpha > 0$$

Properties of the Gamma Function:

- $\Gamma(1) = 1$
- For $\alpha > 1$, $\Gamma(\alpha) = (\alpha - 1)\Gamma(\alpha - 1)$
- If n is an integer and $n > 1$, then $\Gamma(n) = (n-1)!$

Density Function:

$$f(x) = \frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} e^{-x/\beta}, \quad x > 0$$

CDF (for $\alpha = n$, and $x > 0$ a positive integer):

$$F(x, n, \beta) = 1 - \sum_{k=0}^{n-1} \frac{(\frac{x}{\beta})^k e^{-x/\beta}}{k!}$$

PDF:

```
hist(
    rgamma(n = 1000, shape = 5, rate = 1),
    main = "Histogram of Gamma Distribution",
    xlab = "Value",
    ylab = "Frequency"
)
```

MGF

$$m_X(t) = (1 - \beta t)^{-\alpha}, \quad t < \frac{1}{\beta}$$

Mean

$$\mu = E[X] = \alpha\beta$$

Variance

$$\sigma^2 = \mathrm{Var}(X) = \alpha\beta^2$$

2.2.8.3.3 Normal Distribution The normal distribution, denoted as $N(\mu, \sigma^2)$, is symmetric and bell-shaped with parameters μ (mean) and σ^2 (variance). It is also known as the Gaussian distribution.

Density Function:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \quad -\infty < x < \infty,\ \sigma > 0$$

CDF: Use table or numerical methods.

PDF

```r
hist(
    rnorm(1000, mean = 0, sd = 1),
    main = "Histogram of Normal Distribution",
    xlab = "Value",
    ylab = "Frequency"
)
```

MGF

$$m_X(t) = e^{\mu t + \frac{\sigma^2 t^2}{2}}$$

Mean

$$\mu = E[X]$$

Variance

$$\sigma^2 = \mathrm{Var}(X)$$

Standard Normal Random Variable:

- The normal random variable Z with mean $\mu = 0$ and standard deviation $\sigma = 1$ is called a standard normal random variable.

- Any normal random variable X with mean μ and standard deviation σ can be converted to the standard normal random variable Z:

$$Z = \frac{X - \mu}{\sigma}$$

Normal Approximation to the Binomial Distribution:

Let X be binomial with parameters n and p. For large n:

- If $p \leq 0.5$ and $np > 5$, or
- If $p > 0.5$ and $n(1-p) > 5$,

X is approximately normally distributed with mean $\mu = np$ and standard deviation $\sigma = \sqrt{np(1-p)}$.

When using the normal approximation, add or subtract 0.5 as needed for the continuity correction.

Discrete Approximate Normal (Corrected):

Table 2.4: **Normal Probability Rule**

Discrete	Approximate Normal (corrected)
$P(X = c)$	$P(c - 0.5 < Y < c + 0.5)$
$P(X < c)$	$P(Y < c - 0.5)$
$P(X \leq c)$	$P(Y < c + 0.5)$
$P(X > c)$	$P(Y > c + 0.5)$
$P(X \geq c)$	$P(Y > c - 0.5)$

If X is normally distributed with parameters μ and σ, then

- $P(-\sigma < X - \mu < \sigma) \approx .68$
- $P(-2\sigma < X - \mu < 2\sigma) \approx .95$
- $P(-3\sigma < X - \mu < 3\sigma) \approx .997$

2.2.8.3.4 Logistic Distribution The logistic distribution is a continuous probability distribution commonly used in logistic regression and other types of statistical modeling. It resembles the normal distribution but has heavier tails, allowing for more extreme values. - The logistic distribution is symmetric around μ. - Its heavier tails make it useful for modeling outcomes with occasional extreme values.

Density Function

$$f(x;\mu,s) = \frac{e^{-(x-\mu)/s}}{s\left(1+e^{-(x-\mu)/s}\right)^2}, \quad -\infty < x < \infty$$

where μ is the location parameter (mean) and $s > 0$ is the scale parameter.

CDF

$$F(x;\mu,s) = \frac{1}{1+e^{-(x-\mu)/s}}, \quad -\infty < x < \infty$$

PDF

```
hist(
    rlogis(1000, location = 0, scale = 1),
    main = "Histogram of Logistic Distribution",
    xlab = "Value",
    ylab = "Frequency"
)
```

MGF

The MGF of the logistic distribution does not exist because its expected value diverges for most t.

Mean

$$\mu = E[X] = \mu$$

Variance

$$\sigma^2 = \mathrm{Var}(X) = \frac{\pi^2 s^2}{3}$$

2.2.8.3.5 Laplace Distribution The Laplace distribution, also known as the double exponential distribution, is a continuous probability distribution often used in economics, finance, and engineering. It is characterized by a peak at its mean and heavier tails compared to the normal distribution.

- The Laplace distribution is symmetric around μ.
- It has heavier tails than the normal distribution, making it suitable for modeling data with more extreme outliers.

Density Function

$$f(x; \mu, b) = \frac{1}{2b} e^{-|x-\mu|/b}, \quad -\infty < x < \infty$$

where μ is the location parameter (mean) and $b > 0$ is the scale parameter.

CDF

$$F(x; \mu, b) = \begin{cases} \frac{1}{2} e^{(x-\mu)/b} & \text{if } x < \mu \\ 1 - \frac{1}{2} e^{-(x-\mu)/b} & \text{if } x \geq \mu \end{cases}$$

PDF

```r
hist(
    VGAM::rlaplace(1000, location = 0, scale = 1),
    main = "Histogram of Laplace Distribution",
    xlab = "Value",
    ylab = "Frequency"
)
```

MGF

$$m_X(t) = \frac{e^{\mu t}}{1 - b^2 t^2}, \quad |t| < \frac{1}{b}$$

Mean

$$\mu = E[X] = \mu$$

Variance

$$\sigma^2 = \mathrm{Var}(X) = 2b^2$$

2.2.8.3.6 Log-normal Distribution The log-normal distribution is denoted as Lognormal(μ, σ^2).

PDF

```
hist(rlnorm(n = 1000, meanlog = 0, sdlog = 1), main="Histogram of Log-normal Distributi
```

2.2.8.3.7 Lognormal Distribution The lognormal distribution is a continuous probability distribution of a random variable whose logarithm is normally distributed. It is often used to model variables that are positively skewed, such as income or biological measurements.

- The lognormal distribution is positively skewed.
- It is useful for modeling data that cannot take negative values and is often used in finance and environmental studies.

Density Function

$$f(x; \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-(\ln(x)-\mu)^2/(2\sigma^2)}, \quad x > 0$$

where μ is the mean of the underlying normal distribution and $\sigma > 0$ is the standard deviation.

CDF

The cumulative distribution function of the lognormal distribution is given by:

$$F(x; \mu, \sigma) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{\ln(x) - \mu}{\sigma\sqrt{2}}\right)\right], \quad x > 0$$

PDF

```
hist(
    rlnorm(1000, meanlog = 0, sdlog = 1),
```

```
    main = "Histogram of Lognormal Distribution",
    xlab = "Value",
    ylab = "Frequency"
)
```

MGF

The moment generating function (MGF) of the lognormal distribution does not exist in a simple closed form.

Mean

$$E[X] = e^{\mu+\sigma^2/2}$$

Variance

$$\sigma^2 = \mathrm{Var}(X) = \left(e^{\sigma^2} - 1\right) e^{2\mu+\sigma^2}$$

2.2.8.3.8 Exponential Distribution The exponential distribution, denoted as $\mathrm{Exp}(\lambda)$, is a special case of the gamma distribution with $\alpha = 1$.

- It is commonly used to model the time between independent events that occur at a constant rate. It is often applied in reliability analysis and queuing theory.
- The exponential distribution is memoryless, meaning the probability of an event occurring in the future is independent of the past.

- It is commonly used to model waiting times, such as the time until the next customer arrives or the time until a radioactive particle decays.

Density Function

$$f(x) = \frac{1}{\beta} e^{-x/\beta}, \quad x, \beta > 0$$

CDF

$$F(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 - e^{-x/\beta} & \text{if } x > 0 \end{cases}$$

PDF

```r
hist(rexp(n = 1000, rate = 1),
     main = "Histogram of Exponential Distribution",
     xlab = "Value",
     ylab = "Frequency")
```

MGF

$$m_X(t) = (1 - \beta t)^{-1}, \quad t < 1/\beta$$

Mean

$$\mu = E[X] = \beta$$

Variance

$$\sigma^2 = \text{Var}(X) = \beta^2$$

2.2.8.3.9 Chi-Squared Distribution The chi-squared distribution is a continuous probability distribution commonly used in statistical inference, particularly in hypothesis testing and construction of confidence intervals for variance. It is also used in goodness-of-fit tests.

- The chi-squared distribution is defined only for positive values.
- It is often used to model the distribution of the sum of the squares of k independent standard normal random variables.

Density Function

$$f(x;k) = \frac{1}{2^{k/2}\Gamma(k/2)} x^{k/2-1} e^{-x/2}, \quad x \geq 0$$

where k is the degrees of freedom and Γ is the gamma function.

CDF

The cumulative distribution function of the chi-squared distribution is given by:

$$F(x;k) = \frac{\gamma(k/2, x/2)}{\Gamma(k/2)}, \quad x \geq 0$$

where γ is the lower incomplete gamma function.

PDF

```r
hist(
    rchisq(1000, df = 5),
    main = "Histogram of Chi-Squared Distribution",
    xlab = "Value",
    ylab = "Frequency"
)
```

MGF

$$m_X(t) = (1-2t)^{-k/2}, \quad t < \frac{1}{2}$$

Mean

$$E[X] = k$$

Variance

$$\sigma^2 = \mathrm{Var}(X) = 2k$$

2.2.8.3.10 Student's T Distribution The **Student's t-distribution** is named after **William Sealy Gosset**, a statistician at Guinness Brewery in the early 20th century. Gosset developed the t-distribution to address small-sample problems in quality control. Since Guinness prohibited employees from publishing under their names, Gosset used the pseudonym **"Student"** when he published his work in 1908 (Student, 1908). The name has stuck ever since, honoring his contribution to statistics.

The Student's t-distribution, denoted as $T(v)$, is defined by:

$$T = \frac{Z}{\sqrt{\chi_v^2/v}},$$

where Z is a standard normal random variable and χ_v^2 follows a chi-squared distribution with v degrees of freedom.

The Student's T distribution is a continuous probability distribution used in statistical inference, particularly for estimating population parameters when the sample size is small and/or the population variance is unknown. It is similar to the normal distribution but has heavier tails, which makes it more robust for small sample sizes.

- The Student's T distribution is symmetric around 0.
- It has heavier tails than the normal distribution, making it useful for dealing with outliers or small sample sizes.

Density Function

$$f(x; v) = \frac{\Gamma((v+1)/2)}{\sqrt{v\pi}\Gamma(v/2)} \left(1 + \frac{x^2}{v}\right)^{-(v+1)/2}$$

where v is the degrees of freedom and $\Gamma(x)$ is the Gamma function.

CDF

The cumulative distribution function of the Student's T distribution is more complex and typically evaluated using numerical methods.

PDF

```r
hist(
    rt(1000, df = 5),
    main = "Histogram of Student's T Distribution",
    xlab = "Value",
    ylab = "Frequency"
)
```

MGF

The moment generating function (MGF) of the Student's T distribution does not exist in a simple closed form.

Mean

For $v > 1$:

$$E[X] = 0$$

Variance

For $v > 2$:

$$\sigma^2 = \mathrm{Var}(X) = \frac{v}{v-2}$$

2.2.8.3.11 Non-central T Distribution The non-central t-distribution, denoted as $T(v, \lambda)$, is a generalization of the Student's t-distribution. It is defined as:

$$T = \frac{Z + \lambda}{\sqrt{\chi_v^2 / v}},$$

where Z is a standard normal random variable, χ_v^2 follows a chi-squared distribution with v degrees of freedom, and λ is the **non-centrality parameter**. This additional parameter introduces asymmetry to the distribution.

The non-central t-distribution arises in scenarios where the null hypothesis does not hold, such as under the alternative hypothesis in hypothesis testing. The non-centrality parameter λ represents the degree to which the mean deviates from zero.

- For $\lambda = 0$, the non-central t-distribution reduces to the Student's t-distribution.
- The distribution is skewed for $\lambda \neq 0$, with the skewness increasing as λ grows.

Density Function

The density function of the non-central t-distribution is more complex and depends on v and λ. It can be expressed in terms of an infinite sum:

$$f(x; v, \lambda) = \sum_{k=0}^{\infty} \frac{e^{-\lambda^2/2}(\lambda^2/2)^k}{k!} \cdot \frac{\Gamma((v+k+1)/2)}{\sqrt{v\pi}\Gamma((v+k)/2)} \left(1 + \frac{x^2}{v}\right)^{-(v+k+1)/2}.$$

PDF

```r
n <- 100  # Number of samples
df <- 5    # Degrees of freedom
lambda <- 2  # Non-centrality parameter


hist(
  rt(n, df = df, ncp = lambda),
  main = "Histogram of Non-central T Distribution",
  xlab = "Value",
  ylab = "Frequency"
)
```

CDF

The cumulative distribution function of the non-central t-distribution is typically computed using numerical methods due to its complexity.

Mean

For $v > 1$:

$$E[T] = \lambda\sqrt{\frac{v}{2}} \cdot \frac{\Gamma((v-1)/2)}{\Gamma(v/2)}.$$

Variance

For $v > 2$:

$$\mathrm{Var}(T) = \frac{v}{v-2} + \lambda^2.$$

Comparison: Student's T vs. Non-central T

Feature	Student's t-distribution	Non-central t-distribution
Definition	$T = \frac{Z}{\sqrt{\chi_v^2/v}}$	$T = \frac{Z+\lambda}{\sqrt{\chi_v^2/v}}$
Centered at	0	λ
Symmetry	Symmetric	Skewed for $\lambda \neq 0$
Parameters	Degrees of freedom (v)	v and λ

Feature	Student's t-distribution	Non-central t-distribution
Shape as $v \to \infty$ (df $\to \infty$)	Normal(0, 1)	Normal(λ, 1)
Applications	Hypothesis testing under null	Power analysis, alternative testing

While the Student's t-distribution is used for standard hypothesis testing and confidence intervals, the non-central t-distribution finds its applications in scenarios involving non-null hypotheses, such as power and sample size calculations.

2.2.8.3.12 F Distribution The F-distribution, denoted as $F(d_1, d_2)$, is strictly positive and used to compare variances.

Definition:

$$F = \frac{\chi^2_{d_1}/d_1}{\chi^2_{d_2}/d_2},$$

where $\chi^2_{d_1}$ and $\chi^2_{d_2}$ are independent chi-squared random variables with degrees of freedom d_1 and d_2, respectively.

The distribution is asymmetric and never negative.

The F distribution arises frequently as the null distribution of a test statistic, especially in the context of comparing variances, such as in analysis of variance (ANOVA).

Density Function

$$f(x; d_1, d_2) = \frac{\sqrt{\frac{(d_1 x)^{d_1} d_2^{d_2}}{(d_1 x + d_2)^{d_1 + d_2}}}}{x B\left(\frac{d_1}{2}, \frac{d_2}{2}\right)}, \quad x > 0$$

where d_1 and d_2 are the degrees of freedom and B is the beta function.

CDF

The cumulative distribution function of the F distribution is typically evaluated using numerical methods.

PDF

```
hist(
    rf(1000, df1 = 5, df2 = 2),
    main = "Histogram of F Distribution",
    xlab = "Value",
    ylab = "Frequency"
)
```

MGF

The moment generating function (MGF) of the F distribution does not exist in a simple closed form.

Mean

For $d_2 > 2$:

$$E[X] = \frac{d_2}{d_2 - 2}$$

Variance

For $d_2 > 4$:

$$\sigma^2 = \mathrm{Var}(X) = \frac{2d_2^2(d_1 + d_2 - 2)}{d_1(d_2 - 2)^2(d_2 - 4)}$$

2.2.8.3.13 Cauchy Distribution The Cauchy distribution is a continuous probability distribution that is often used in physics and has heavier tails than the normal distribution. It is notable because it does not have a finite mean or variance.

- The Cauchy distribution does not have a finite mean or variance.
- The Central Limit Theorem and Weak Law of Large Numbers do not apply to the Cauchy distribution.

Density Function

$$f(x; x_0, \gamma) = \frac{1}{\pi\gamma\left[1 + \left(\frac{x-x_0}{\gamma}\right)^2\right]}$$

where x_0 is the location parameter and $\gamma > 0$ is the scale parameter.

CDF

The cumulative distribution function of the Cauchy distribution is given by:

$$F(x; x_0, \gamma) = \frac{1}{\pi}\arctan\left(\frac{x - x_0}{\gamma}\right) + \frac{1}{2}$$

PDF

```
hist(
    rcauchy(1000, location = 0, scale = 1),
    main = "Histogram of Cauchy Distribution",
    xlab = "Value",
    ylab = "Frequency"
)
```

MGF

The MGF of the Cauchy distribution does not exist.

Mean

The mean of the Cauchy distribution is undefined.

Variance

The variance of the Cauchy distribution is undefined.

2.2.8.3.14 Multivariate Normal Distribution Let y be a p-dimensional multivariate normal (MVN) random variable with mean μ and variance-covariance matrix Σ. The density function of y is given by:

$$f(\mathbf{y}) = \frac{1}{(2\pi)^{p/2}|\ |^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{y}-\mu)'\Sigma^{-1}(\mathbf{y}-\mu)\right)$$

where $|\ |$ represents the determinant of the variance-covariance matrix Σ, and $\mathbf{y} \sim N_p(\ ,\ )$.

Properties:

- Let $\mathbf{A}_{r\times p}$ be a fixed matrix. Then $\mathbf{Ay} \sim N_r(\mathbf{A}\ , \mathbf{A}\ \mathbf{A}')$. Note that $r \leq p$, and all rows of $\mathbf{A}$ must be linearly independent to guarantee that $\mathbf{A}\ \mathbf{A}'$ is non-singular.
- Let $\mathbf{G}$ be a matrix such that $^{-\mathbf{1}} = \mathbf{GG}'$. Then $\mathbf{G}'\mathbf{y} \sim N_p(\mathbf{G}'\ , \mathbf{I})$ and $\mathbf{G}'(\mathbf{y} -\ ) \sim N_p(\mathbf{0}, \mathbf{I})$.
- Any fixed linear combination of $y_1, \ldots, y_p$, say $\mathbf{c}'\mathbf{y}$, follows $\mathbf{c}'\mathbf{y} \sim N_1(\mathbf{c}'\ , \mathbf{c}'\ \mathbf{c})$.

Large Sample Properties

Suppose that $y_1, \ldots, y_n$ are a random sample from some population with mean μ and variance-covariance matrix Σ:

$$\mathbf{Y} \sim MVN(\ ,\ )$$

Then:

- $\bar{\mathbf{y}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_i$ is a consistent estimator for .
- $\mathbf{S} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})'$ is a consistent estimator for .
- **Multivariate Central Limit Theorem**: Similar to the univariate case, $\sqrt{n}(\bar{\mathbf{y}} - \mu) \sim N_p(\mathbf{0},\ )$ when n is large relative to p (e.g., $n \geq 25p$), which is equivalent to $\bar{\mathbf{y}} \sim N_p(\ ,\ /\mathbf{n})$.
- **Wald's Theorem**: $n(\bar{\mathbf{y}} - \mu)'\mathbf{S}^{-\mathbf{1}}(\bar{\mathbf{y}} - \mu) \sim \chi^2_{(p)}$ when n is large relative to p.

Density Function

$$f(\mathbf{x}; \mu, \Sigma) = \frac{1}{(2\pi)^{k/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$

where μ is the mean vector, Σ is the covariance matrix, $\mathbf{x} \in \mathbb{R}^k$ and k is the number of variables.

CDF

The cumulative distribution function of the multivariate normal distribution does not have a simple closed form and is typically evaluated using numerical methods.

PDF

```
k <- 2
n <- 1000
mu <- c(0, 0)
sigma <- matrix(c(1, 0.5, 0.5, 1), nrow = k)
library(MASS)
hist(
    mvrnorm(n, mu = mu, Sigma = sigma)[,1],
    main = "Histogram of MVN Distribution (1st Var)",
    xlab = "Value",
    ylab = "Frequency"
)
```

MGF

$$m_{\mathbf{X}}(\mathbf{t}) = \exp\left(\mu^T \mathbf{t} + \frac{1}{2}\mathbf{t}^T \Sigma \mathbf{t}\right)$$

Mean

$$E[\mathbf{X}] = \mu$$

Variance

$$\mathrm{Var}(\mathbf{X}) = \Sigma$$

2.3 General Math

2.3.1 Number Sets

Notation	Denotes	Examples
$\emptyset$	Empty set	No members
$\mathbb{N}$	Natural numbers	$\{1, 2, ...\}$
$\mathbb{Z}$	Integers	$\{..., -1, 0, 1, ...\}$
$\mathbb{Q}$	Rational numbers	Including fractions
$\mathbb{R}$	Real numbers	Including all finite decimals, irrational numbers
$\mathbb{C}$	Complex numbers	Including numbers of the form $a + bi$ where $i^2 = -1$

2.3.2 Summation Notation and Series

2.3.2.1 Chebyshev's Inequality

Let X be a random variable with mean μ and standard deviation σ. For any positive number k, Chebyshev's Inequality states:

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

This provides a probabilistic bound on the deviation of X from its mean and does not require X to follow a normal distribution.

2.3.2.2 Geometric Sum

For a geometric series of the form $\sum_{k=0}^{n-1} ar^k$, the sum is given by:

$$\sum_{k=0}^{n-1} ar^k = a\frac{1-r^n}{1-r} \quad \text{where } r \neq 1$$

2.3.2.3 Infinite Geometric Series

When $|r| < 1$, the geometric series converges to:

$$\sum_{k=0}^{\infty} ar^k = \frac{a}{1-r}$$

2.3.2.4 Binomial Theorem

The binomial expansion for $(x+y)^n$ is:

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k \quad \text{where } n \geq 0$$

2.3.2.5 Binomial Series

For non-integer exponents α:

$$\sum_{k=0}^{\infty} \binom{\alpha}{k} x^k = (1+x)^\alpha \quad \text{where } |x| < 1$$

2.3.2.6 Telescoping Sum

A telescoping sum simplifies as intermediate terms cancel, leaving:

$$\sum_{a \leq k < b} \Delta F(k) = F(b) - F(a) \quad \text{where } a, b \in \mathbb{Z}, a \leq b$$

2.3.2.7 Vandermonde Convolution

The Vandermonde convolution identity is:

$$\sum_{k=0}^{n} \binom{r}{k} \binom{s}{n-k} = \binom{r+s}{n} \quad \text{where } n \in \mathbb{Z}$$

2.3.2.8 Exponential Series

The exponential function e^x can be represented as:

$$\sum_{k=0}^{\infty} \frac{x^k}{k!} = e^x \quad \text{where } x \in \mathbb{C}$$

2.3.2.9 Taylor Series

The Taylor series expansion for a function $f(x)$ about $x = a$ is:

$$\sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k = f(x)$$

For $a = 0$, this becomes the **Maclaurin series**.

2.3.2.10 Maclaurin Series for e^z

A special case of the Taylor series, the Maclaurin expansion for e^z is:

$$e^z = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \cdots$$

2.3.2.11 Euler's Summation Formula

Euler's summation formula connects sums and integrals:

$$\sum_{a \le k < b} f(k) = \int_a^b f(x)\,dx + \sum_{k=1}^{m} \frac{B_k}{k!}\left[f^{(k-1)}(x)\right]_a^b + (-1)^{m+1} \int_a^b \frac{B_m(x - \lfloor x \rfloor)}{m!} f^{(m)}(x)\,dx$$

Here, B_k are Bernoulli numbers.

- **For** $m = 1$ (Trapezoidal Rule):

$$\sum_{a \le k < b} f(k) \approx \int_a^b f(x)\,dx - \frac{1}{2}(f(b) - f(a))$$

2.3.3 Taylor Expansion

A differentiable function, $G(x)$, can be written as an infinite sum of its derivatives. More specifically, if $G(x)$ is infinitely differentiable and evaluated at a, its Taylor expansion is:

$$G(x) = G(a) + \frac{G'(a)}{1!}(x-a) + \frac{G''(a)}{2!}(x-a)^2 + \frac{G'''(a)}{3!}(x-a)^3 + ...$$

This expansion is valid within the radius of convergence.

2.3.4 Law of Large Numbers

Let $X_1, X_2, ...$ be an infinite sequence of independent and identically distributed (i.i.d.) random variables with finite mean μ and variance σ^2. The **Law of Large Numbers (LLN)** states that the sample average:

$$\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$$

converges to the expected value μ as $n \to \infty$. This can be expressed as:

$$\bar{X}_n \to \mu \quad (\text{as } n \to \infty).$$

2.3.4.1 Variance of the Sample Mean

The variance of the sample mean decreases as the sample size increases:

$$Var(\bar{X}_n) = Var\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{\sigma^2}{n}.$$

$$\begin{aligned} Var(\bar{X}_n) &= Var(\frac{1}{n}(X_1 + ... + X_n)) = Var\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) \\ &= \frac{1}{n^2}Var(X_1 + ... + X_n) \\ &= \frac{n\sigma^2}{n^2} = \frac{\sigma^2}{n} \end{aligned}$$

Note: The connection between the Law of Large Numbers and the Normal Distribution lies in the Central Limit Theorem. The CLT states that, regardless of the original distribution of a dataset, the distribution of the sample means will tend to follow a normal distribution as the sample size becomes larger.

The difference between [Weak Law] and [Strong Law] regards the mode of convergence.

2.3.4.2 Weak Law of Large Numbers

The **Weak Law of Large Numbers** states that the sample average converges in probability to the expected value:

$$\bar{X}_n \xrightarrow{p} \mu \quad \text{as } n \to \infty.$$

Formally, for any $\epsilon > 0$:

$$\lim_{n\to\infty} P(|\bar{X}_n - \mu| > \epsilon) = 0.$$

Additionally, the sample mean of an i.i.d. random sample ($\{X_i\}_{i=1}^n$) from any population with a finite mean and variance is a consistent estimator of the population mean μ:

$$plim(\bar{X}_n) = plim\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \mu.$$

2.3.4.3 Strong Law of Large Numbers

The **Strong Law of Large Numbers** states that the sample average converges almost surely to the expected value:

$$\bar{X}_n \xrightarrow{a.s.} \mu \quad \text{as } n \to \infty.$$

Equivalently, this can be expressed as:

$$P\left(\lim_{n\to\infty} \bar{X}_n = \mu\right) = 1.$$

2.3.5 Convergence

2.3.5.1 Convergence in Probability

As $n \to \infty$, an estimator (random variable) θ_n is said to converge in probability to a constant c if:

$$\lim_{n\to\infty} P(|\theta_n - c| \geq \epsilon) = 0 \quad \text{for any } \epsilon > 0.$$

This is denoted as:

$$plim(\theta_n) = c \quad \text{or equivalently, } \theta_n \xrightarrow{p} c.$$

Properties of Convergence in Probability:

1. **Slutsky's Theorem**: For a continuous function $g(\cdot)$, if $plim(\theta_n) = \theta$, then:

$$plim(g(\theta_n)) = g(\theta)$$

2. If $\gamma_n \xrightarrow{p} \gamma$, then:
   - $plim(\theta_n + \gamma_n) = \theta + \gamma$,
   - $plim(\theta_n \gamma_n) = \theta\gamma$,
   - $plim(\theta_n/\gamma_n) = \theta/\gamma$ (if $\gamma \neq 0$).
3. These properties extend to random vectors and matrices.

2.3.5.2 Convergence in Distribution

As $n \to \infty$, the distribution of a random variable X_n may converge to another ("fixed") distribution. Formally, X_n with CDF $F_n(x)$ converges in distribution to X with CDF $F(x)$ if:

$$\lim_{n\to\infty} |F_n(x) - F(x)| = 0$$

at all points of continuity of $F(x)$. This is denoted as:

$$X_n \xrightarrow{d} X \quad \text{or equivalently, } F(x) \text{ is the limiting distribution of } X_n.$$

Asymptotic Properties:

- $E(X)$: Limiting mean (asymptotic mean).
- $Var(X)$: Limiting variance (asymptotic variance).

Note: Limiting expectations and variances do not necessarily match the expectations and variances of X_n:

$$E(X) \neq \lim_{n\to\infty} E(X_n),$$
$$Avar(X_n) \neq \lim_{n\to\infty} Var(X_n).$$

Properties of Convergence in Distribution:

1. **Continuous Mapping Theorem**: For a continuous function $g(\cdot)$, if $X_n \xrightarrow{d} X$, then:

$$g(X_n) \xrightarrow{d} g(X).$$

2. If $Y_n \xrightarrow{d} c$ (a constant), then:
   - $X_n + Y_n \xrightarrow{d} X + c$,
   - $Y_n X_n \xrightarrow{d} cX$,
   - $X_n/Y_n \xrightarrow{d} X/c$ (if $c \neq 0$).
3. These properties also extend to random vectors and matrices.

2.3.5.3 Summary: Properties of Convergence

Convergence in Probability	Convergence in Distribution
Slutsky's Theorem: For a continuous $g(\cdot)$, if $plim(\theta_n) = \theta$, then $plim(g(\theta_n)) = g(\theta)$	Continuous Mapping Theorem: For a continuous $g(\cdot)$, if $X_n \xrightarrow{d} X$, then $g(X_n) \xrightarrow{d} g(X)$
If $\gamma_n \xrightarrow{p} \gamma$, then:	If $Y_n \xrightarrow{d} c$, then:
$plim(\theta_n + \gamma_n) = \theta + \gamma$	$X_n + Y_n \xrightarrow{d} X + c$
$plim(\theta_n \gamma_n) = \theta\gamma$	$Y_n X_n \xrightarrow{d} cX$
$plim(\theta_n/\gamma_n) = \theta/\gamma$ (if $\gamma \neq 0$)	$X_n/Y_n \xrightarrow{d} X/c$ (if $c \neq 0$)

Relationship between Convergence Types:

Convergence in Probability is stronger than Convergence in Distribution. Therefore:

- Convergence in Distribution does not guarantee Convergence in Probability.

2.3.6 Sufficient Statistics and Likelihood

2.3.6.1 Likelihood

The **likelihood** describes the degree to which the observed data supports a particular value of a parameter θ.

- The exact value of the likelihood is **not meaningful**; only relative comparisons matter.
- Likelihood is **informative** when comparing parameter values, helping identify which values of θ are more plausible given the data.

For a single observation $Y = y$, the likelihood function is defined as:

$$L(\theta_0; y) = P(Y = y \mid \theta = \theta_0) = f_Y(y; \theta_0),$$

where $f_Y(y; \theta_0)$ is the probability density (or mass) function of Y for the parameter θ_0.

Key Insight: The likelihood tells us how plausible θ is, given the data we observed. It is **not a probability**, but it is proportional to the probability of observing the data under a given parameter value.

Example: Suppose Y follows a binomial distribution with $n = 10$ trials and probability of success p:

$$P(Y = y \mid p) = \binom{10}{y} p^y (1-p)^{10-y}.$$

For $y = 7$ observed successes, the likelihood function becomes:

$$L(p; y = 7) = \binom{10}{7} p^7 (1-p)^3.$$

We can use this to compare how well different values of p explain the observed data.

2.3.6.2 Likelihood Ratio

The **likelihood ratio** compares the relative likelihood of two parameter values θ_0 and θ_1 given the observed data:

$$\text{Likelihood Ratio} = \frac{L(\theta_0; y)}{L(\theta_1; y)}.$$

- A likelihood ratio greater than 1 implies that θ_0 is more likely than θ_1, given the observed data.

- Likelihood ratios are widely used in hypothesis testing and model comparison to evaluate the evidence against a null hypothesis.

Example: For the binomial example above, consider $p_0 = 0.7$ and $p_1 = 0.5$. The likelihood ratio is:

$$\frac{L(p_0; y = 7)}{L(p_1; y = 7)} = \frac{\binom{10}{7}(0.7)^7(0.3)^3}{\binom{10}{7}(0.5)^7(0.5)^3}.$$

This simplifies to:

$$\frac{(0.7)^7(0.3)^3}{(0.5)^7(0.5)^3}.$$

The likelihood ratio quantifies how much more likely p_0 is compared to p_1 given the observed data.

2.3.6.3 Likelihood Function

For a given sample, the likelihood for all possible values of θ forms the **likelihood function**:

$$L(\theta) = L(\theta; y) = f_Y(y; \theta).$$

For a sample of size n, assuming independence among observations:

$$L(\theta) = \prod_{i=1}^{n} f_Y(y_i; \theta).$$

Taking the natural logarithm of the likelihood gives the **log-likelihood function**:

$$l(\theta) = \sum_{i=1}^{n} \log f_Y(y_i; \theta).$$

Why Log-Likelihood?

- The log-likelihood simplifies computation by turning products into sums.
- It is particularly useful for optimization, as many numerical methods (e.g., gradient-based algorithms) perform better with sums than products.

Example: For $Y_1, Y_2, \ldots, Y_n$ i.i.d. observations from a normal distribution $N(\mu, \sigma^2)$, the likelihood is:

$$L(\mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mu)^2}{2\sigma^2}\right).$$

The log-likelihood is:

$$l(\mu, \sigma^2) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mu)^2.$$

2.3.6.4 Sufficient Statistics

A **sufficient statistic** $T(y)$ is a summary of the data that retains all information about a parameter θ. It allows us to focus on this condensed statistic without losing any inferential power regarding θ.

Formal Definition:

A statistic $T(y)$ is sufficient for a parameter θ if the conditional probability distribution of the data y, given $T(y)$ and θ, does not depend on θ. Mathematically:

$$P(Y = y \mid T(y), \theta) = P(Y = y \mid T(y)).$$

Alternatively, by the **Factorization Theorem**, $T(y)$ is sufficient if the likelihood can be written as:

$$L(\theta; y) = c(y)L^*(\theta; T(y)),$$

where:

- $c(y)$ is a function of the data independent of θ.
- $L^*(\theta; T(y))$ is a function that depends on θ and $T(y)$.

In other words, the likelihood function can be rewritten in terms of $T(y)$ alone, without loss of information about θ.

Why Sufficient Statistics Matter:

- They allow us to simplify the analysis by reducing the data without losing inferential power.
- Many inferential procedures (e.g., Maximum Likelihood Estimation, Bayesian methods) are simplified by working with sufficient statistics.

Example:

Consider a sample of i.i.d. observations $Y_1, Y_2, \dots, Y_n$ from a normal distribution $N(\mu, \sigma^2)$. Here:

1. The sample mean $\bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$ is sufficient for μ.
2. The sample variance $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \bar{Y})^2$ is sufficient for σ^2.

Verification: The joint density of $y_1, y_2, \dots, y_n$ can be factored as:

$$f(y_1, \dots, y_n; \mu, \sigma^2) = \underbrace{\frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \bar{y})^2\right)}_{L^*(\mu,\sigma^2;\bar{y},s^2)} \cdot \underbrace{[\text{independent of } \mu,\, \sigma^2]}_{c(y)}.$$

This shows $\bar{Y}$ and S^2 are sufficient.

Usage of Sufficient Statistics

1. **Maximum Likelihood Estimation (MLE):** In MLE, sufficient statistics simplify the optimization problem by reducing the data without losing information.

   Example: In the normal distribution case, μ can be estimated using the sufficient statistic $\bar{Y}$:
   $$\hat{\mu}_{MLE} = \bar{Y}.$$

2. **Bayesian Inference:** In Bayesian analysis, the posterior distribution depends on the sufficient statistic rather than the entire data set. For the normal case:
   $$P(\mu \mid \bar{Y}) \propto P(\mu)L(\mu; \bar{Y}).$$

3. **Data Compression:** In practice, sufficient statistics reduce the complexity of data storage and analysis by condensing all relevant information into a smaller representation.

2.3.6.5 Nuisance Parameters

Parameters that are not of direct interest in the analysis but are necessary to model the data are called **nuisance parameters**.

Profile Likelihood: To handle nuisance parameters, replace them with their maximum likelihood estimates (MLEs) in the likelihood function, creating a **profile likelihood** for the parameter of interest.

Example of Profile Likelihood:

In a regression model with parameters β (coefficients) and σ^2 (error variance), σ^2 is often a nuisance parameter. The profile likelihood for β is obtained by substituting the MLE of σ^2 into the likelihood:

$$L_p(\beta) = L(\beta, \hat{\sigma}^2),$$

where $\hat{\sigma}^2$ is the MLE of σ^2 given β.

This simplifies the problem to focus only on the parameter of interest, β.

2.3.7 Parameter Transformations

Transformations of parameters are often used to improve interpretability or statistical properties of models.

2.3.7.1 Log-Odds Transformation

The **log-odds transformation** is commonly used in logistic regression and binary classification problems. It transforms probabilities (which are bounded between 0 and 1) to the real line:

$$\text{Log odds} = g(\theta) = \ln\left(\frac{\theta}{1-\theta}\right),$$

where θ represents a probability (e.g., the success probability in a Bernoulli trial).

2.3.7.2 General Parameter Transformations

For a parameter θ and a transformation $g(\cdot)$:

- If $\theta \in (a, b)$, $g(\theta)$ may map θ to a different range (e.g., $\mathbb{R}$).
- Useful transformations include:
  - Logarithmic: $g(\theta) = \ln(\theta)$ for $\theta > 0$.
  - Exponential: $g(\theta) = e^\theta$ for unconstrained θ.
  - Square root: $g(\theta) = \sqrt{\theta}$ for $\theta \geq 0$.

Jacobian Adjustment for Transformations: If transforming a parameter in Bayesian inference, the Jacobian of the transformation must be included to ensure proper posterior scaling.

2.3.7.3 Applications of Parameter Transformations

1. **Improving Interpretability**:
   - Probabilities can be transformed to odds or log-odds for logistic models.
   - Rates can be transformed logarithmically for multiplicative effects.
2. **Statistical Modeling**:
   - Variance-stabilizing transformations (e.g., log for Poisson data or arcsine for proportions).
   - Regularization or simplification of complex relationships.
3. **Optimization**:
   - Transforming constrained parameters (e.g., probabilities or positive scales) to unconstrained scales simplifies optimization algorithms.

2.4 Data Import/Export

Extended Manual by R

Table 2.8: Table by Rio Vignette

Format	Typical Extension	Import Package	Export Package	Installed by Default
Comma-separated data	.csv	**data.table**	**data.table**	Yes
Pipe-separated data	.psv	**data.table**	**data.table**	Yes
Tab-separated data	.tsv	**data.table**	**data.table**	Yes
CSVY (CSV + YAML metadata header)	.csvy	**data.table**	**data.table**	Yes
SAS	.sas7bdat	**haven**	**haven**	Yes
SPSS	.sav	**haven**	**haven**	Yes
SPSS (compressed)	.zsav	**haven**	**haven**	Yes
Stata	.dta	**haven**	**haven**	Yes
SAS XPORT	.xpt	**haven**	**haven**	Yes
SPSS Portable	.por	**haven**		Yes
Excel	.xls	**readxl**		Yes
Excel	.xlsx	**readxl**	**openxlsx**	Yes

Format	Typical Extension	Import Package	Export Package	Installed by Default
R syntax	.R	**base**	**base**	Yes
Saved R objects	.RData, .rda	**base**	**base**	Yes
Serialized R objects	.rds	**base**	**base**	Yes
Epiinfo	.rec	**foreign**		Yes
Minitab	.mtp	**foreign**		Yes
Systat	.syd	**foreign**		Yes
"XBASE" database files	.dbf	**foreign**	**foreign**	Yes
Weka Attribute-Relation File Format	.arff	**foreign**	**foreign**	Yes
Data Interchange Format	.dif	**utils**		Yes
Fortran data	no recognized extension	**utils**		Yes
Fixed-width format data	.fwf	**utils**	**utils**	Yes
gzip comma-separated data	.csv.gz	**utils**	**utils**	Yes
Apache Arrow (Parquet)	.parquet	**arrow**	**arrow**	No
EViews	.wf1	**hexView**		No
Feather R/Python interchange format	.feather	**feather**	**feather**	No
Fast Storage	.fst	**fst**	**fst**	No
JSON	.json	**jsonlite**	**jsonlite**	No
Matlab	.mat	**rmatio**	**rmatio**	No
OpenDocument Spreadsheet	.ods	**readODS**	**readODS**	No
HTML Tables	.html	**xml2**	**xml2**	No
Shallow XML documents	.xml	**xml2**	**xml2**	No
YAML	.yml	**yaml**	**yaml**	No
Clipboard	default is tsv	**clipr**	**clipr**	No

Format	Typical Extension	Import Package	Export Package	Installed by Default
Google Sheets	as Comma-separated data			

R limitations:

- By default, R use 1 core in CPU
- R puts data into memory (limit around 2-4 GB), while SAS uses data from files on demand
- Categorization
  - Medium-size file: within RAM limit, around 1-2 GB
  - Large file: 2-10 GB, there might be some workaround solution
  - Very large file > 10 GB, you have to use distributed or parallel computing

Solutions:

- buy more RAM
- HPC packages
  - Explicit Parallelism
  - Implicit Parallelism
  - Large Memory
  - Map/Reduce
- specify number of rows and columns, typically including command `nrow =`
- Use packages that store data differently
  - `bigmemory`, `biganalytics`, `bigtabulate` , `synchronicity`, `bigalgebra`, `bigvideo` use C++ to store matrices, but also support one class type
  - For multiple class types, use `ff` package
- Very Large datasets use
  - `RHaddop` package
  - `HadoopStreaming`
  - `Rhipe`

2.4.1 Medium size

```
library("rio")
```

To import multiple files in a directory

```
str(import_list(dir()), which = 1)
```

To export a single data file

```
export(data, "data.csv")
export(data,"data.dta")
export(data,"data.txt")
export(data,"data_cyl.rds")
export(data,"data.rdata")
export(data,"data.R")
export(data,"data.csv.zip")
export(data,"list.json")
```

To export multiple data files

```
export(list(mtcars = mtcars, iris = iris), "data_file_type")
# where data_file_type should substituted with the extension listed above
```

To convert between data file types

```
# convert Stata to SPSS
convert("data.dta", "data.sav")
```

2.4.2 Large size

2.4.2.1 Cloud Computing: Using AWS for Big Data

Amazon Web Service (AWS): Compute resources can be rented at approximately $1/hr. Use AWS to process large datasets without overwhelming your local machine.

2.4.2.2 Importing Large Files as Chunks

```
file_in <- file("in.csv", "r")  # Open a connection to the file
chunk_size <- 100000            # Define chunk size
x <- readLines(file_in, n = chunk_size)  # Read data in chunks
close(file_in)                  # Close the file connection
```

2.4.2.2.1 Using Base R

```r
library(data.table)
mydata <- fread("in.csv", header = TRUE)  # Fast and memory-efficient
```

2.4.2.2.2 Using the `data.table` Package

```r
library(ff)
x <- read.csv.ffdf(
  file = "file.csv",
  nrow = 10,          # Total rows
  header = TRUE,      # Include headers
  VERBOSE = TRUE,     # Display progress
  first.rows = 10000, # Initial chunk
  next.rows = 50000,  # Subsequent chunks
  colClasses = NA
)
```

2.4.2.2.3 Using the `ff` Package

```r
library(bigmemory)
my_data <- read.big.matrix('in.csv', header = TRUE)
```

2.4.2.2.4 Using the `bigmemory` Package

```r
library(sqldf)
my_data <- read.csv.sql('in.csv')

# Example: Filtering during import
iris2 <- read.csv.sql("iris.csv",
    sql = "SELECT * FROM file WHERE Species = 'setosa'")
```

2.4.2.2.5 Using the `sqldf` Package

```r
library(RMySQL)
```

2.4.2.2.6 Using the `RMySQL` Package `RQLite` package

- Download SQLite, pick "A bundle of command-line tools for managing SQLite database files" for Window 10
- Unzip file, and open `sqlite3.exe`.
- Type in the prompt
  - `sqlite> .cd 'C:\Users\data'` specify path to your desired directory
  - `sqlite> .open database_name.db` to open a database

- To import the CSV file into the database
  - `sqlite> .mode csv` specify to SQLite that the next file is .csv file
  - `sqlite> .import file_name.csv datbase_name` to import the csv file to the database
- `sqlite> .exit` After you're done, exit the sqlite program

```
library(DBI)
library(dplyr)
library("RSQLite")
setwd("")
con <- dbConnect(RSQLite::SQLite(), "data_base.db")
tbl <- tbl(con, "data_table")
tbl %>%
    filter() %>%
    select() %>%
    collect() # to actually pull the data into the workspace
dbDisconnect(con)
```

```
library(arrow)
data <- read_csv_arrow("file.csv")
```

2.4.2.2.7 Using the `arrow` Package

```
library(vroom)

# Import a compressed CSV file
compressed <- vroom_example("mtcars.csv.zip")
data <- vroom(compressed)
```

2.4.2.2.8 Using the `vroom` Package

```
s = fread("sample.csv")
```

2.4.2.2.9 Using the `data.table` Package

```
test = ff::read.csv.ffdf(file = "")
object.size(test) # Highest memory usage

test1 = data.table::fread(file = "")
object.size(test1) # Lowest memory usage
```

```
test2 = readr::read_csv(file = "")
object.size(test2) # Second lowest memory usage

test3 = vroom::vroom(file = "")
object.size(test3) # Similar to read_csv
```

2.4.2.2.10 Comparisons Regarding Storage Space To work with large datasets, you can compress them into `csv.gz` format. However, typically, R requires loading the entire dataset before exporting it, which can be impractical for data over 10 GB. In such cases, processing the data sequentially becomes necessary. Although `read.csv` is slower compared to `readr::read_csv`, it can handle connections and allows for sequential looping, making it useful for large files.

Currently, `readr::read_csv` does not support the `skip` argument efficiently for large data. Even if you specify `skip`, the function reads all preceding lines again. For instance, if you run `read_csv(file, n_max = 100, skip = 0)` followed by `read_csv(file, n_max = 200, skip = 100)`, the first 100 rows are re-read. In contrast, `read.csv` can continue from where it left off without re-reading previous rows.

If you encounter an error such as:

"Error in (function (con, what, n = 1L, size = NA_integer_, signed = TRUE): can only read from a binary connection",

you can modify the connection mode from `"r"` to `"rb"` (read binary). Although the `file` function is designed to detect the appropriate format automatically, this workaround can help resolve the issue when it does not behave as expected.

2.4.2.3 Sequential Processing for Large Data

```
# Open file for sequential reading
file_conn <- file("file.csv", open = "r")
while (TRUE) {
  # Read a chunk of data
  data_chunk <- read.csv(file_conn, nrows = 1000)
  if (nrow(data_chunk) == 0) break  # Stop if no more rows
  # Process the chunk here
}
close(file_conn)  # Close connection
```

2.5 Data Manipulation

```
# Load required packages
library(tidyverse)
```

```r
library(lubridate)

# -----------------------------
# Data Structures in R
# -----------------------------

# Create vectors
x <- c(1, 4, 23, 4, 45)
n <- c(1, 3, 5)
g <- c("M", "M", "F")

# Create a data frame
df <- data.frame(n, g)
df  # View the data frame
#>   n g
#> 1 1 M
#> 2 3 M
#> 3 5 F
str(df)  # Check its structure
#> 'data.frame':    3 obs. of  2 variables:
#>  $ n: num  1 3 5
#>  $ g: chr  "M" "M" "F"

# Using tibble for cleaner outputs
df <- tibble(n, g)
df  # View the tibble
#> # A tibble: 3 x 2
#>       n g
#>   <dbl> <chr>
#> 1     1 M
#> 2     3 M
#> 3     5 F
str(df)
#> tibble [3 x 2] (S3: tbl_df/tbl/data.frame)
#>  $ n: num [1:3] 1 3 5
#>  $ g: chr [1:3] "M" "M" "F"

# Create a list
lst <- list(x, n, g, df)
lst  # Display the list
#> [[1]]
#> [1]  1  4 23  4 45
#>
#> [[2]]
#> [1] 1 3 5
```

```
#>
#> [[3]]
#> [1] "M" "M" "F"
#>
#> [[4]]
#> # A tibble: 3 x 2
#>       n g
#>   <dbl> <chr>
#> 1     1 M
#> 2     3 M
#> 3     5 F

# Name list elements
lst2 <- list(num = x, size = n, sex = g, data = df)
lst2  # Named list elements are easier to reference
#> $num
#> [1]  1  4 23  4 45
#>
#> $size
#> [1] 1 3 5
#>
#> $sex
#> [1] "M" "M" "F"
#>
#> $data
#> # A tibble: 3 x 2
#>       n g
#>   <dbl> <chr>
#> 1     1 M
#> 2     3 M
#> 3     5 F

# Another list example with numeric vectors
lst3 <- list(
  x = c(1, 3, 5, 7),
  y = c(2, 2, 2, 4, 5, 5, 5, 6),
  z = c(22, 3, 3, 3, 5, 10)
)
lst3
#> $x
#> [1] 1 3 5 7
#>
#> $y
#> [1] 2 2 2 4 5 5 5 6
#>
```

```
#> $z
#> [1] 22  3  3  3  5 10

# Find means of list elements
# One at a time
mean(lst3$x)
#> [1] 4
mean(lst3$y)
#> [1] 3.875
mean(lst3$z)
#> [1] 7.666667

# Using lapply to calculate means
lapply(lst3, mean)
#> $x
#> [1] 4
#>
#> $y
#> [1] 3.875
#>
#> $z
#> [1] 7.666667

# Simplified output with sapply
sapply(lst3, mean)
#>        x        y        z
#> 4.000000 3.875000 7.666667

# Tidyverse alternative: map() function
map(lst3, mean)
#> $x
#> [1] 4
#>
#> $y
#> [1] 3.875
#>
#> $z
#> [1] 7.666667

# Tidyverse with numeric output: map_dbl()
map_dbl(lst3, mean)
#>        x        y        z
#> 4.000000 3.875000 7.666667

# ----------------------------
```

```r
# Binding Data Frames
# ----------------------------

# Create tibbles for demonstration
dat01 <- tibble(x = 1:5, y = 5:1)
dat02 <- tibble(x = 10:16, y = x / 2)
dat03 <- tibble(z = runif(5))  # 5 random numbers from (0, 1)

# Row binding
bind_rows(dat01, dat02, dat01)
#> # A tibble: 17 x 2
#>        x     y
#>    <int> <dbl>
#>  1     1   5
#>  2     2   4
#>  3     3   3
#>  4     4   2
#>  5     5   1
#>  6    10   5
#>  7    11   5.5
#>  8    12   6
#>  9    13   6.5
#> 10    14   7
#> 11    15   7.5
#> 12    16   8
#> 13     1   5
#> 14     2   4
#> 15     3   3
#> 16     4   2
#> 17     5   1

# Add a new identifier column with .id
bind_rows(dat01, dat02, .id = "id")
#> # A tibble: 12 x 3
#>    id        x     y
#>    <chr> <int> <dbl>
#>  1 1         1   5
#>  2 1         2   4
#>  3 1         3   3
#>  4 1         4   2
#>  5 1         5   1
#>  6 2        10   5
#>  7 2        11   5.5
#>  8 2        12   6
#>  9 2        13   6.5
```

```
#> 10 2        14   7
#> 11 2        15   7.5
#> 12 2        16   8

# Use named inputs for better identification
bind_rows("dat01" = dat01, "dat02" = dat02, .id = "id")
#> # A tibble: 12 x 3
#>    id        x     y
#>    <chr> <int> <dbl>
#>  1 dat01     1   5
#>  2 dat01     2   4
#>  3 dat01     3   3
#>  4 dat01     4   2
#>  5 dat01     5   1
#>  6 dat02    10   5
#>  7 dat02    11   5.5
#>  8 dat02    12   6
#>  9 dat02    13   6.5
#> 10 dat02    14   7
#> 11 dat02    15   7.5
#> 12 dat02    16   8

# Bind a list of data frames
list01 <- list("dat01" = dat01, "dat02" = dat02)
bind_rows(list01, .id = "source")
#> # A tibble: 12 x 3
#>    source     x     y
#>    <chr>  <int> <dbl>
#>  1 dat01      1   5
#>  2 dat01      2   4
#>  3 dat01      3   3
#>  4 dat01      4   2
#>  5 dat01      5   1
#>  6 dat02     10   5
#>  7 dat02     11   5.5
#>  8 dat02     12   6
#>  9 dat02     13   6.5
#> 10 dat02     14   7
#> 11 dat02     15   7.5
#> 12 dat02     16   8

# Column binding
bind_cols(dat01, dat03)
#> # A tibble: 5 x 3
#>       x     y      z
```

```
#>   <int> <int>  <dbl>
#> 1     1     5 0.508
#> 2     2     4 0.824
#> 3     3     3 0.0374
#> 4     4     2 0.577
#> 5     5     1 0.988

# ------------------------------
# String Manipulation
# ------------------------------

names <- c("Ford, MS", "Jones, PhD", "Martin, Phd", "Huck, MA, MLS")

# Remove everything after the first comma
str_remove(names, pattern = ", [[:print:]]+")
#> [1] "Ford"   "Jones"  "Martin" "Huck"

# Explanation: [[:print:]]+ matches one or more printable characters

# ------------------------------
# Reshaping Data
# ------------------------------

# Wide format data
wide <- data.frame(
  name = c("Clay", "Garrett", "Addison"),
  test1 = c(78, 93, 90),
  test2 = c(87, 91, 97),
  test3 = c(88, 99, 91)
)

# Long format data
long <- data.frame(
  name = rep(c("Clay", "Garrett", "Addison"), each = 3),
  test = rep(1:3, 3),
  score = c(78, 87, 88, 93, 91, 99, 90, 97, 91)
)

# Summary statistics
aggregate(score ~ name, data = long, mean)  # Mean score per student
#>      name    score
#> 1 Addison 92.66667
#> 2    Clay 84.33333
#> 3 Garrett 94.33333
aggregate(score ~ test, data = long, mean)  # Mean score per test
```

```
#>   test    score
#> 1    1 87.00000
#> 2    2 91.66667
#> 3    3 92.66667

# Line plot of scores over tests
ggplot(long,
       aes(
           x = factor(test),
           y = score,
           color = name,
           group = name
       )) +
    geom_point() +
    geom_line() +
    xlab("Test") +
    ggtitle("Test Scores by Student")
```

```
# Reshape wide to long
pivot_longer(wide, test1:test3, names_to = "test", values_to = "score")
#> # A tibble: 9 x 3
#>   name    test  score
#>   <chr>   <chr> <dbl>
#> 1 Clay    test1    78
#> 2 Clay    test2    87
#> 3 Clay    test3    88
```

```
#> 4 Garrett test1    93
#> 5 Garrett test2    91
#> 6 Garrett test3    99
#> 7 Addison test1    90
#> 8 Addison test2    97
#> 9 Addison test3    91

# Use names_prefix to clean column names
pivot_longer(
    wide,
    -name,
    names_to = "test",
    values_to = "score",
    names_prefix = "test"
)
#> # A tibble: 9 x 3
#>   name    test  score
#>   <chr>   <chr> <dbl>
#> 1 Clay    1        78
#> 2 Clay    2        87
#> 3 Clay    3        88
#> 4 Garrett 1        93
#> 5 Garrett 2        91
#> 6 Garrett 3        99
#> 7 Addison 1        90
#> 8 Addison 2        97
#> 9 Addison 3        91

# Reshape long to wide with explicit id_cols argument
pivot_wider(
  long,
  id_cols = name,
  names_from = test,
  values_from = score
)
#> # A tibble: 3 x 4
#>   name      `1`   `2`   `3`
#>   <chr>   <dbl> <dbl> <dbl>
#> 1 Clay       78    87    88
#> 2 Garrett    93    91    99
#> 3 Addison    90    97    91

# Add a prefix to the resulting columns
pivot_wider(
  long,
```

```
  id_cols = name,
  names_from = test,
  values_from = score,
  names_prefix = "test"
)
#> # A tibble: 3 x 4
#>   name    test1 test2 test3
#>   <chr>   <dbl> <dbl> <dbl>
#> 1 Clay       78    87    88
#> 2 Garrett    93    91    99
#> 3 Addison    90    97    91
```

The verbs of data manipulation

- `select`: selecting (or not selecting) columns based on their names (eg: select columns Q1 through Q25)
- `slice`: selecting (or not selecting) rows based on their position (eg: select rows 1:10)
- `mutate`: add or derive new columns (or variables) based on existing columns (eg: create a new column that expresses measurement in cm based on existing measure in inches)
- `rename`: rename variables or change column names (eg: change "GraduationRate100" to "grad100")
- `filter`: selecting rows based on a condition (eg: all rows where gender = Male)
- `arrange`: ordering rows based on variable(s) numeric or alphabetical order (eg: sort in descending order of Income)
- `sample`: take random samples of data (eg: sample 80% of data to create a "training" set)
- `summarize`: condense or aggregate multiple values into single summary values (eg: calculate median income by age group)
- `group_by`: convert a tbl into a grouped tbl so that operations are performed "by group"; allows us to summarize data or apply verbs to data by groups (eg, by gender or treatment)
- the pipe: `%>%`
  - Use Ctrl + Shift + M (Win) or Cmd + Shift + M (Mac) to enter in RStudio
  - The pipe takes the output of a function and "pipes" into the first argument of the next function.
  - new pipe is `|>` It should be identical to the old one, except for certain special cases.
- `:=` (Walrus operator): similar to `=` , but for cases where you want to use the `glue` package (i.e., dynamic changes in the variable name in the left-hand side)

Writing function in R

Tunneling

`{{` (called curly-curly) allows you to tunnel data-variables through arg-variables (i.e., function arguments)

```
library(tidyverse)
# ------------------------------
# Writing Functions with {{ }}
# ------------------------------

# Define a custom function using {{ }}
get_mean <- function(data, group_var, var_to_mean) {
  data %>%
    group_by({{group_var}}) %>%
    summarize(mean = mean({{var_to_mean}}, na.rm = TRUE))
}

# Apply the function
data("mtcars")
mtcars %>%
  get_mean(group_var = cyl, var_to_mean = mpg)
#> # A tibble: 3 x 2
#>     cyl  mean
#>   <dbl> <dbl>
#> 1     4  26.7
#> 2     6  19.7
#> 3     8  15.1

# Dynamically name the resulting variable
get_mean <- function(data, group_var, var_to_mean, prefix = "mean_of") {
  data %>%
    group_by({{group_var}}) %>%
    summarize("{prefix}_{{var_to_mean}}" := mean({{var_to_mean}}, na.rm = TRUE))
}

# Apply the modified function
mtcars %>%
  get_mean(group_var = cyl, var_to_mean = mpg)
#> # A tibble: 3 x 2
#>     cyl mean_of_mpg
#>   <dbl>       <dbl>
#> 1     4        26.7
#> 2     6        19.7
#> 3     8        15.1
```

I. BASIC

Chapter 3

Descriptive Statistics

When you have an area of interest to research, a problem to solve, or a relationship to investigate, theoretical and empirical processes will help you.

Estimand: Defined as "a quantity of scientific interest that can be calculated in the population and does not change its value depending on the data collection design used to measure it (i.e., it does not vary with sample size, survey design, the number of non-respondents, or follow-up efforts)." (Rubin, 1996)

Examples of estimands include:

- Population means
- Population variances
- Correlations
- Factor loadings
- Regression coefficients

3.1 Numerical Measures

There are differences between a population and a sample:

Measures of	Category	Population	Sample
	What is it?	Reality	A small fraction of reality (inference)
	Characteristics described by	Parameters	Statistics

Measures of	Category	Population	Sample
Central Tendency	Mean	$\mu = E(Y)$	$\hat{\mu} = \overline{y}$
Central Tendency	Median	50th percentile	$y_{(\frac{n+1}{2})}$
Dispersion	Variance	$\sigma^2 = var(Y) = E[(Y-\mu)^2]$	$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \overline{y})^2$
Dispersion	Coefficient of Variation	$\frac{\sigma}{\mu}$	$\frac{s}{\overline{y}}$
Dispersion	Interquartile Range	Difference between 25th and 75th percentiles; robust to outliers	
Shape	Skewness Standardized 3rd central moment (unitless)	$g_1 = \frac{\mu_3}{\sigma^3}$	$\hat{g_1} = \frac{m_3}{m_2^{3/2}}$
Shape	Central moments	$\mu = E(Y)$, $\mu_2 = \sigma^2 = E[(Y-\mu)^2]$, $\mu_3 = E[(Y-\mu)^3]$, $\mu_4 = E[(Y-\mu)^4]$	$m_2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2$ $m_3 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^3$
Shape	Kurtosis (peakedness and tail thickness) Standardized 4th central moment	$g_2^* = \frac{E[(Y-\mu)^4]}{\sigma^4}$	$\hat{g_2} = \frac{m_4}{m_2^2} - 3$

Notes:

1. **Order Statistics**: $y_{(1)}, y_{(2)}, \ldots, y_{(n)}$, where $y_{(1)} < y_{(2)} < \ldots < y_{(n)}$.
2. **Coefficient of Variation**:
   - Defined as the standard deviation divided by the mean.
   - A stable, unitless statistic useful for comparison.
3. **Symmetry**:
   - **Symmetric distributions**: Mean = Median; Skewness = 0.
   - **Skewed Right**: Mean > Median; Skewness > 0.
   - **Skewed Left**: Mean < Median; Skewness < 0.
4. **Central Moments**:
   - $\mu = E(Y)$

- $\mu_2 = \sigma^2 = E[(Y-\mu)^2]$
- $\mu_3 = E[(Y-\mu)^3]$
- $\mu_4 = E[(Y-\mu)^4]$

Skewness ($\hat{g_1}$)

1. **Sampling Distribution**:
   For samples drawn from a normal population:
   - $\hat{g_1}$ is approximately distributed as $N(0, \frac{6}{n})$ when $n > 150$.
2. **Inference**:
   - **Large Samples**: Inference on skewness can be based on the standard normal distribution.
     The 95% confidence interval for g_1 is given by:

     $$\hat{g_1} \pm 1.96\sqrt{\frac{6}{n}}$$

   - **Small Samples**: For small samples, consult special tables such as:
     - Snedecor and Cochran (1989), Table A 19(i)
     - Monte Carlo test results

Kurtosis ($\hat{g_2}$)

1. **Definitions and Relationships**:
   - A normal distribution has kurtosis $g_2^* = 3$.
     Kurtosis is often redefined as:

     $$g_2 = \frac{E[(Y-\mu)^4]}{\sigma^4} - 3$$

     where the 4th central moment is estimated by:

     $$m_4 = \frac{\sum_{i=1}^{n}(y_i - \overline{y})^4}{n}$$

2. **Sampling Distribution**:
   For large samples ($n > 1000$):
   - $\hat{g_2}$ is approximately distributed as $N(0, \frac{24}{n})$.
3. **Inference**:
   - **Large Samples**: Inference for kurtosis can use standard normal tables.
   - **Small Samples**: Refer to specialized tables such as:
     - Snedecor and Cochran (1989), Table A 19(ii)
     - Geary (1936)

Kurtosis Value	Tail Behavior	Comparison to Normal Distribution
$g_2 > 0$ (Leptokurtic)	Heavier Tails	Examples: t-distributions
$g_2 < 0$ (Platykurtic)	Lighter Tails	Examples: Uniform or certain bounded distributions
$g_2 = 0$ (Mesokurtic)	Normal Tails	Exactly matches the normal distribution

```r
# Generate random data from a normal distribution
data <- rnorm(100)

# Load the e1071 package for skewness and kurtosis functions
library(e1071)

# Calculate skewness
skewness_value <- skewness(data)
cat("Skewness:", skewness_value, "\n")
#> Skewness: 0.1003872

# Calculate kurtosis
kurtosis_value <- kurtosis(data)
cat("Kurtosis:", kurtosis_value, "\n")
#> Kurtosis: -0.5484067
```

3.2 Graphical Measures

The following table summarizes key graphical measures along with guidance on when and why to use each. More detailed explanations, visual examples, and sample code will be discussed after this table.

Graph Type	When to Use	Why It's Useful
Histogram	- Exploring the distribution (shape, center, spread) of a single continuous variable	- Quickly identifies frequency, modes, skewness, and potential outliers - Provides an overview of data "shape"
Box-and-Whisker Plot	- Comparing the same continuous variable across multiple categories - Identifying median, IQR, and outliers	- Shows distribution at a glance (median, quartiles) - Highlights outliers and potential group differences

Graph Type	When to Use	Why It's Useful
Stem-and-Leaf Plot	- Small, single-variable datasets where you want a textual yet visual distribution view	- Reveals the distribution while preserving actual data values - Easy to spot clusters and gaps for small datasets
Notched Boxplot	- Similar to a standard boxplot but with confidence intervals around the median	- If notches don't overlap, it suggests the medians differ significantly - Helps clarify whether differences in medians are likely meaningful
Bagplot (2D Box-plot)	- Bivariate data where you want a 2D "boxplot"-style overview - Identifying outliers in two-dimensional space	- Depicts both central region ("bag") and potential outliers - Ideal for discovering clusters or unusual points in two continuous variables
Boxplot Matrix	- Multiple continuous variables that you want to compare side-by-side	- Quickly compares distributions of many variables simultaneously - Helpful for spotting differences in median, spread, and outliers
Violin Plot	- Same use case as boxplot but you want *more detail* on the distribution's shape	- Combines boxplot features with a density plot - Shows where data are concentrated or sparse within each category
Scatterplot	- Two continuous variables to check for relationships, trends, or outliers	- Visualizes correlation or non-linear patterns - Aids in identifying clusters or extreme values
Pairwise Scatter-plots	- Initial exploration of *several* variables at once	- Enables a quick scan of relationships between *all* variable pairs - Useful for identifying multivariate patterns or potential correlation structures

Tips for Selecting the Right Plot:

- **Focus on Your Question**: Are you comparing groups, investigating correlations, or just exploring the overall shape of the data?

- **Match the Plot to Your Data Type**: Continuous vs. categorical data often dictates your choice of chart.
- **Mind the Data Size**: Some plots become cluttered or lose clarity with very large datasets (e.g., stem-and-leaf), while others may be less informative with very few data points.

3.2.1 Shape

Properly labeling your graphs is essential to ensure that viewers can easily understand the data presented. Below are several examples of graphical measures used to assess the shape of a dataset.

```r
# Generate random data for demonstration purposes
data <- rnorm(100)

# Histogram: A graphical representation of the distribution of a dataset.
hist(
    data,
    labels = TRUE,
    col = "grey",
    breaks = 12,
    main = "Histogram of Random Data",
    xlab = "Value",
    ylab = "Frequency"
)
```

```r
# Interactive Histogram: Using 'highcharter' for a more interactive visualization.
```

```r
# pacman::p_load("highcharter")
# hchart(data, type = "column", name = "Random Data Distribution")

# Box-and-Whisker Plot: Useful for visualizing the distribution and identifying outliers.
boxplot(
    count ~ spray,
    data = InsectSprays,
    col = "lightgray",
    main = "Boxplot of Insect Sprays",
    xlab = "Spray Type",
    ylab = "Count"
)
```

```r
# Notched Boxplot: The notches indicate a confidence interval around the median.
boxplot(
    len ~ supp * dose,
    data = ToothGrowth,
    notch = TRUE,
    col = c("gold", "darkgreen"),
    main = "Tooth Growth by Supplement and Dose",
    xlab = "Supplement and Dose",
    ylab = "Length"
)
```

```
# If the notches of two boxes do not overlap, this suggests that the medians differ si

# Stem-and-Leaf Plot: Provides a quick way to visualize the distribution of data.
stem(data)
#>
#>   The decimal point is at the |
#>
#>   -2 | 4300
#>   -1 | 8644432221110
#>   -0 | 988888776666643322221100 0
#>    0 | 00000011111122222333344444455666677778899
#>    1 | 0000111223455889
#>    2 | 28

# Bagplot - A 2D Boxplot Extension: Visualizes the spread and identifies outliers in t
pacman::p_load(aplpack)
attach(mtcars)
bagplot(wt,
        mpg,
        xlab = "Car Weight",
        ylab = "Miles Per Gallon",
        main = "Bagplot of Car Weight vs. Miles Per Gallon")
```

```
detach(mtcars)
```

Below are some advanced plot types that can provide deeper insights into data:

```
# boxplot.matrix(): Creates boxplots for each column in a matrix. Useful for comparing multiple v
graphics::boxplot.matrix(
    cbind(
        Uni05 = (1:100) / 21,
        Norm = rnorm(100),
        T5 = rt(100, df = 5),
        Gam2 = rgamma(100, shape = 2)
    ),
    main = "Boxplot Marix",
    notch = TRUE,
    col = 1:4
)
```

```
# Violin Plot (vioplot()): Combines a boxplot with a density plot, providing more info
library("vioplot")
vioplot(data, col = "lightblue", main = "Violin Plot Example")
```

3.2.2 Scatterplot

Scatterplots are useful for visualizing relationships between two continuous variables. They help identify patterns, correlations, and outliers.

Pairwise Scatterplots: Visualizes relationships between all pairs of variables in a dataset. This is especially useful for exploring potential correlations.

```
pairs(mtcars,
      main = "Pairwise Scatterplots",
      pch = 19,
      col = "blue")
```

3.3 Normality Assessment

The Normal (Gaussian) distribution plays a critical role in statistical analyses due to its theoretical and practical applications. Many statistical methods assume normality in the data, making it essential to assess whether our variable of interest follows a normal distribution. To achieve this, we utilize both Numerical Measures and Graphical Assessment.

3.3.1 Graphical Assessment

Graphical methods provide an intuitive way to visually inspect the normality of a dataset. One of the most common methods is the **Q-Q plot** (quantile-quantile plot). The Q-Q plot compares the quantiles of the sample data to the quantiles of a theoretical normal distribution. Deviations from the line indicate departures from normality.

Below is an example of using the `qqnorm` and `qqline` functions in R to assess the normality of the `precip` dataset, which contains precipitation data (in inches per year) for 70 U.S. cities:

```r
# Load the required package
pacman::p_load("car")

# Generate a Q-Q plot
qqnorm(precip,
       ylab = "Precipitation [in/yr] for 70 US cities",
       main = "Q-Q Plot of Precipitation Data")
qqline(precip, col = "red")
```

Interpretation

- **Theoretical Line**: The red line represents the expected relationship if the data were perfectly normally distributed.
- **Data Points**: The dots represent the actual empirical data.

If the points closely align with the theoretical line, we can conclude that the data likely follow a normal distribution. However, noticeable deviations from the line, particularly systematic patterns (e.g., curves or s-shaped patterns), indicate potential departures from normality.

Tips

1. **Small Deviations**: Minor deviations from the line in small datasets are not uncommon and may not significantly impact analyses that assume normality.
2. **Systematic Patterns**: Look for clear trends, such as clusters or s-shaped curves, which suggest skewness or heavy tails.

3. **Complementary Tests**: Always pair graphical methods with numerical measures (e.g., Shapiro-Wilk test) to make a robust conclusion.

When interpreting a Q-Q plot, it is helpful to see both ideal and non-ideal scenarios. Below is an illustrative example:

1. **Normal Data**: Points fall closely along the line.
2. **Skewed Data**: Points systematically deviate from the line, curving upward or downward.
3. **Heavy Tails**: Points deviate at the extremes (ends) of the distribution.

By combining visual inspection and numerical measures, we can better understand the nature of our data and its alignment with the assumption of normality.

3.3.2 Summary Statistics

While graphical assessments, such as Q-Q plots, provide a visual indication of normality, they may not always offer a definitive conclusion. To supplement graphical methods, statistical tests are often employed. These tests provide quantitative evidence to support or refute the assumption of normality. The most common methods can be classified into two categories:

- Methods Based on Normal Probability Plot
  - Correlation Coefficient with Normal Probability Plots
  - Shapiro-Wilk Test
- Methods based on empirical cumulative distribution function
  - Anderson-Darling Test
  - Kolmogorov-Smirnov Test
  - Cramer-von Mises Test
  - Jarque–Bera Test

3.3.2.1 Methods Based on Normal Probability Plot

3.3.2.1.1 Correlation Coefficient with Normal Probability Plots As described by Looney and Gulledge Jr (1985) and Shapiro and Francia (1972), this method evaluates the linearity of a normal probability plot by calculating the correlation coefficient between the ordered sample values $y_{(i)}$ and their theoretical normal quantiles m_i^*. A perfectly linear relationship suggests that the data follow a normal distribution.

The correlation coefficient, denoted W^*, is given by:

$$W^* = \frac{\sum_{i=1}^{n}(y_{(i)} - \bar{y})(m_i^* - 0)}{\sqrt{\sum_{i=1}^{n}(y_{(i)} - \bar{y})^2 \cdot \sum_{i=1}^{n}(m_i^* - 0)^2}}$$

where:

- $\bar{y}$ is the sample mean,
- $\bar{m}^* = 0$ under the null hypothesis of normality.

The **Pearson product-moment correlation formula** can also be used to evaluate this relationship:

$$\hat{\rho} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2 \cdot \sum_{i=1}^{n}(x_i - \bar{x})^2}}$$

- **Interpretation**:
  - When the correlation is 1, the plot is exactly linear, and normality is assumed.
  - The closer the correlation is to 0, the stronger the evidence to reject normality.
  - Inference on W^* requires reference to special tables (Looney and Gulledge Jr, 1985).

```r
library("EnvStats")

# Perform Probability Plot Correlation Coefficient (PPCC) Test
gofTest(data, test = "ppcc")$p.value # Probability Plot Correlation Coefficient
#> [1] 0.7094038
```

3.3.2.1.2 Shapiro-Wilk Test The Shapiro-Wilk test (Shapiro and Wilk, 1965) is one of the most widely used tests for assessing normality, especially for sample sizes $n < 2000$. This test evaluates how well the data's order statistics match a theoretical normal distribution. The test statistic, W, is computed as:

$$W = \frac{\sum_{i=1}^{n} a_i x_{(i)}}{\sum_{i=1}^{n}(x_{(i)} - \bar{x})^2}$$

where

- n: The sample size.
- $x_{(i)}$: The i-th smallest value in the sample (the ordered data).
- $\bar{x}$: The sample mean.
- a_i: Weights derived from the expected values and variances of the order statistics of a normal distribution, precomputed based on the sample size n.

Sensitive to:

1. **Symmetry**

- The Shapiro-Wilk test assesses whether a sample is drawn from a normal distribution, which assumes **symmetry** around the mean.
- If the data exhibit skewness (a lack of symmetry), the test is likely to reject the null hypothesis of normality.

2. **Heavy Tails**
   - Heavy tails refer to distributions where extreme values (outliers) are more likely compared to a normal distribution.
   - The Shapiro-Wilk test is also sensitive to such departures from normality because heavy tails affect the spread and variance, which are central to the calculation of the test statistic W.

Hence, the Shapiro-Wilk test's sensitivity to these deviations makes it a powerful tool for detecting non-normality only in small to moderate-sized samples. However:

- It is generally more sensitive to **symmetry** (skewness) than to **tail behavior** (kurtosis).
- In very large samples, even small deviations in symmetry or tail behavior may cause the test to reject the null hypothesis, even if the data is practically "normal" for the intended analysis.
  - Small sample sizes may lack power to detect deviations from normality.
  - Large sample sizes may detect minor deviations that are not practically significant.

Key Steps:

1. **Sort the Data:** Arrange the sample data in ascending order, yielding $x_{(1)}, x_{(2)}, \dots, x_{(n)}$.
2. **Compute Weights:** The weights a_i are determined using a covariance matrix of the normal order statistics. These are optimized to maximize the power of the test.
3. **Calculate** W: Use the formula to determine W, which ranges from 0 to 1.

Decision Rule:

- **Null Hypothesis** (H_0): The data follows a normal distribution.
- **Alternative Hypothesis** (H_1): The data does not follow a normal distribution.
- A small W value, along with a p-value below a chosen significance level (e.g., 0.05), leads to rejection of H_0.
  - Under normality, W approaches 1.
  - Smaller values of W indicate deviations from normality.

```r
# Perform Shapiro-Wilk Test (Default for gofTest)
EnvStats::gofTest(mtcars$mpg, test = "sw")
#>
#> Results of Goodness-of-Fit Test
#> -------------------------------
#>
#> Test Method:                     Shapiro-Wilk GOF
#>
#> Hypothesized Distribution:       Normal
#>
#> Estimated Parameter(s):          mean = 20.090625
#>                                  sd   =  6.026948
#>
#> Estimation Method:               mvue
#>
#> Data:                            mtcars$mpg
#>
#> Sample Size:                     32
#>
#> Test Statistic:                  W = 0.9475647
#>
#> Test Statistic Parameter:        n = 32
#>
#> P-value:                         0.1228814
#>
#> Alternative Hypothesis:          True cdf does not equal the
#>                                  Normal Distribution.
```

3.3.2.2 Methods Based on Empirical Cumulative Distribution Function

The **Empirical Cumulative Distribution Function (ECDF)** is a way to represent the distribution of a sample dataset in cumulative terms. It answers the question:

> "What fraction of the observations in my dataset are less than or equal to a given value x?"

The ECDF is defined as:

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(X_i \leq x)$$

where:

- $F_(x)$: ECDF at value x.

- n: Total number of data points.
- $\mathbb{I}(X_i \leq x)$: Indicator function, equal to 1 if $X_i \leq x$, otherwise 0.

This method is especially useful for large sample sizes and can be applied to distributions beyond the normal (Gaussian) distribution.

Properties of the ECDF

1. **Step Function**: The ECDF is a step function that increases by $1/n$ at each data point.
2. **Non-decreasing**: As x increases, $F_n(x)$ never decreases.
3. **Range**: The ECDF starts at 0 and ends at 1:
   - $F_n(x) = 0$ for $x < \min(X)$.
   - $F_n(x) = 1$ for $x \geq \max(X)$.
4. **Convergence**: As $n \to \infty$, the ECDF approaches the true cumulative distribution function (CDF) of the population.

Let's consider a sample dataset $\{3, 7, 7, 10, 15\}$. The ECDF at different values of x is calculated as:

x	$\mathbb{I}(X_i \leq x)$ for each X_i	Count $\leq x$	ECDF $F_n(x)$
$x = 5$	$\{1, 0, 0, 0, 0\}$	1	$1/5 = 0.2$
$x = 7$	$\{1, 1, 1, 0, 0\}$	3	$3/5 = 0.6$
$x = 12$	$\{1, 1, 1, 1, 0\}$	4	$4/5 = 0.8$
$x = 15$	$\{1, 1, 1, 1, 1\}$	5	$5/5 = 1.0$

Applications of the ECDF

1. **Goodness-of-fit Tests**: Compare the ECDF to a theoretical CDF (e.g., using the Kolmogorov-Smirnov test).
2. **Outlier Detection**: Analyze cumulative trends to spot unusual data points.
3. **Visual Data Exploration**: Use the ECDF to understand the spread, skewness, and distribution of the data.
4. **Comparing Distributions**: Compare the ECDFs of two datasets to assess differences in their distributions.

```r
# Load required libraries
library(ggplot2)

# Sample dataset
data <- c(3, 7, 7, 10, 15)

# ECDF calculation
```

```r
ecdf_function <- ecdf(data)

# Generate a data frame for plotting
ecdf_data <- data.frame(x = sort(unique(data)),
                        ecdf = sapply(sort(unique(data)), function(x)
                          mean(data <= x)))

# Display ECDF values
print(ecdf_data)
#>    x ecdf
#> 1  3  0.2
#> 2  7  0.6
#> 3 10  0.8
#> 4 15  1.0

# Plot the ECDF
ggplot(ecdf_data, aes(x = x, y = ecdf)) +
  geom_step() +
  labs(
    title = "Empirical Cumulative Distribution Function",
    x = "Data Values",
    y = "Cumulative Proportion"
  ) +
  theme_minimal()
```

```r
# Alternatively
plot.ecdf(as.numeric(mtcars[1, ]),
```

```
        verticals = TRUE,
        do.points = FALSE)
```

3.3.2.2.1 Anderson-Darling Test The **Anderson-Darling test** statistic (Anderson and Darling, 1952) is given by:

$$A^2 = \int_{-\infty}^{\infty} \frac{\left(F_n(t) - F(t)\right)^2}{F(t)(1 - F(t))} dF(t)$$

This test calculates a weighted average of squared deviations between the empirical cumulative distribution function (CDF), $F_n(t)$, and the theoretical CDF, $F(t)$. More weight is given to deviations in the tails of the distribution, which makes the test particularly sensitive to these regions.

For a sample of size n, with ordered observations $y_{(1)}, y_{(2)}, \dots, y_{(n)}$, the Anderson-Darling test statistic can also be written as:

$$A^2 = -n - \frac{1}{n} \sum_{i=1}^{n} \left[(2i - 1) \ln(F(y_{(i)})) + (2n + 1 - 2i) \ln(1 - F(y_{(i)})) \right]$$

For the **normal distribution**, the test statistic is further simplified. Using the transformation:

$$p_i = \Phi\left(\frac{y_{(i)} - \bar{y}}{s}\right),$$

where:

- p_i is the cumulative probability under the standard normal distribution,
- $y_{(i)}$ are the ordered sample values,
- $\bar{y}$ is the sample mean,
- s is the sample standard deviation,

the formula becomes:

$$A^2 = -n - \frac{1}{n}\sum_{i=1}^{n}\left[(2i-1)\ln(p_i) + (2n+1-2i)\ln(1-p_i)\right].$$

Key Features of the Test

1. **CDF-Based Weighting**: The Anderson-Darling test gives more weight to deviations in the tails, which makes it particularly sensitive to detecting non-normality in these regions.
2. **Sensitivity**: Compared to other goodness-of-fit tests, such as the Kolmogorov-Smirnov Test, the Anderson-Darling test is better at identifying differences in the tails of the distribution.
3. **Integral Form**: The test statistic can also be expressed as an integral over the theoretical CDF:

   $$A^2 = n\int_{-\infty}^{\infty}\frac{\left[F_n(t) - F(t)\right]^2}{F(t)(1-F(t))}dF(t),$$

   where $F_n(t)$ is the empirical CDF, and $F(t)$ is the specified theoretical CDF.
4. **Applications**:
   - Testing for normality or other distributions (e.g., exponential, Weibull).
   - Validating assumptions in statistical models.
   - Comparing data to theoretical distributions.

Hypothesis Testing

- **Null Hypothesis (H_0)**: The data follows the specified distribution (e.g., normal distribution).
- **Alternative Hypothesis (H_1)**: The data does not follow the specified distribution.
- The null hypothesis is rejected if A^2 is too large, indicating a poor fit to the specified distribution.

Critical values for the test statistic are provided by (Marsaglia and Marsaglia, 2004) and (Stephens, 1974).

Applications to Other Distributions

The Anderson-Darling test can be applied to various distributions by using specific transformation methods. Examples include:

- **Exponential**
- **Logistic**
- **Gumbel**
- **Extreme-value**
- **Weibull** (after logarithmic transformation: log(Weibull) = Gumbel)
- **Gamma**
- **Cauchy**
- **von Mises**
- **Log-normal (two-parameter)**

For more details on transformations and critical values, consult (Stephens, 1974).

```
# Perform Anderson-Darling Test
library(nortest)
ad_test_result <- ad.test(mtcars$mpg)

# Output the test statistic and p-value
ad_test_result
#>
#>  Anderson-Darling normality test
#>
#> data:  mtcars$mpg
#> A = 0.57968, p-value = 0.1207
```

Alternatively, for a broader range of distributions, use the `gofTest` function from the `gof` package:

```
# General goodness-of-fit test with Anderson-Darling
library(EnvStats)
gof_test_result <- EnvStats::gofTest(mtcars$mpg, test = "ad")

# Extract the p-value
gof_test_result$p.value
#> [1] 0.1207371
```

3.3.2.2.2 Kolmogorov-Smirnov Test The **Kolmogorov-Smirnov (K-S) test** is a nonparametric test that compares the empirical cumulative distribution function (ECDF) of a sample to a theoretical cumulative distribution

function (CDF), or compares the ECDFs of two samples. It is used to assess whether a sample comes from a specific distribution (one-sample test) or to compare two samples (two-sample test).

The test statistic D_n for the one-sample test is defined as:

$$D_n = \sup_x |F_n(x) - F(x)|,$$

where:

- $F_n(x)$ is the empirical CDF of the sample,
- $F(x)$ is the theoretical CDF under the null hypothesis,
- $\sup_x$ denotes the supremum (largest value) over all possible values of x.

For the two-sample K-S test, the statistic is:

$$D_{n,m} = \sup_x \left|F_{n,1}(x) - F_{m,2}(x)\right|,$$

where $F_{n,1}(x)$ and $F_{m,2}(x)$ are the empirical CDFs of the two samples, with sizes n and m, respectively.

Hypotheses

- **Null hypothesis (H_0)**: The sample comes from the specified distribution (one-sample) or the two samples are drawn from the same distribution (two-sample).
- **Alternative hypothesis (H_1)**: The sample does not come from the specified distribution (one-sample) or the two samples are drawn from different distributions (two-sample).

Properties

1. **Based on the Largest Deviation**: The K-S test is sensitive to the largest absolute difference between the empirical and expected CDFs, making it effective for detecting shifts in location or scale.

2. **Distribution-Free**: The test does not assume a specific distribution for the data under the null hypothesis. Its significance level is determined from the distribution of the test statistic under the null hypothesis.

3. **Limitations**:

   - The test is more sensitive near the center of the distribution than in the tails.
   - It may not perform well with discrete data or small sample sizes.

4. **Related Tests**:

- **Kuiper's Test**: A variation of the K-S test that is sensitive to deviations in both the center and tails of the distribution. The Kuiper test statistic is:

$$V_n = D^+ + D^-,$$

where D^+ and D^- are the maximum positive and negative deviations of the empirical CDF from the theoretical CDF.

Applications

- Testing for normality or other specified distributions.
- Comparing two datasets to determine if they are drawn from the same distribution.

To perform a one-sample K-S test in R, use the `ks.test()` function. To check the goodness of fit for a specific distribution, the `gofTest()` function from a package like `DescTools` can also be used.

```
# One-sample Kolmogorov-Smirnov test for normality
data <- rnorm(50)  # Generate random normal data
ks.test(data, "pnorm", mean(data), sd(data))
#>
#>  Exact one-sample Kolmogorov-Smirnov test
#>
#> data:  data
#> D = 0.095053, p-value = 0.7211
#> alternative hypothesis: two-sided

# Goodness-of-fit test using gofTest
library(DescTools)
gofTest(data, test = "ks")$p.value  # Kolmogorov-Smirnov test p-value
#> [1] 0.7211441
```

- **Advantages**:
  - Simple and widely applicable.
  - Distribution-free under the null hypothesis.
- **Limitations**:
  - Sensitive to sample size: small deviations may lead to significance in large samples.
  - Reduced sensitivity to differences in the tails compared to the Anderson-Darling test.

The Kolmogorov-Smirnov test provides a general-purpose method for goodness-of-fit testing and sample comparison, with particular utility in detecting central deviations.

3.3.2.2.3 Cramer-von Mises Test The **Cramer-von Mises (CVM) test** is a nonparametric goodness-of-fit test that evaluates the agreement between the empirical cumulative distribution function (ECDF) of a sample and a specified theoretical cumulative distribution function (CDF). Unlike the Kolmogorov-Smirnov test, which focuses on the largest discrepancy, the Cramer-von Mises test considers the **average squared discrepancy** across the entire distribution. Unlike the Anderson-Darling test, it weights all parts of the distribution equally.

The test statistic W^2 for the one-sample Cramer-von Mises test is defined as:

$$W^2 = n \int_{-\infty}^{\infty} \left[F_n(t) - F(t)\right]^2 dF(t),$$

where:

- $F_n(t)$ is the empirical CDF,
- $F(t)$ is the specified theoretical CDF under the null hypothesis,
- n is the sample size.

In practice, W^2 is computed using the ordered sample values $y_{(1)}, y_{(2)}, \dots, y_{(n)}$ as:

$$W^2 = \sum_{i=1}^{n} \left(F(y_{(i)}) - \frac{2i-1}{2n}\right)^2 + \frac{1}{12n},$$

where:

- $F(y_{(i)})$ is the theoretical CDF evaluated at the ordered sample values $y_{(i)}$.

Hypotheses

- **Null hypothesis (H0)**: The sample data follow the specified distribution.
- **Alternative hypothesis (H1)**: The sample data do not follow the specified distribution.

Properties

1. **Focus on Average Discrepancy**: The Cramer-von Mises test measures the overall goodness-of-fit by considering the squared deviations across all points in the distribution, ensuring equal weighting of discrepancies.
2. **Comparison to Anderson-Darling**: Unlike the Anderson-Darling test, which gives more weight to deviations in the tails, the CVM test weights all parts of the distribution equally.
3. **Integral Representation**: The statistic is expressed as an integral over the squared differences between the empirical and theoretical CDFs.

4. **Two-Sample Test**: The Cramer-von Mises framework can also be extended to compare two empirical CDFs. The two-sample statistic is based on the pooled empirical CDF.

Applications

- Assessing goodness-of-fit for a theoretical distribution (e.g., normal, exponential, Weibull).
- Comparing two datasets to determine if they are drawn from similar distributions.
- Validating model assumptions.

To perform a Cramer-von Mises test in R, the `gofTest()` function from the `DescTools` package can be used. Below is an example:

```
# Generate random normal data
data <- rnorm(50)

# Perform the Cramer-von Mises test
library(DescTools)
gofTest(data, test = "cvm")$p.value  # Cramer-von Mises test p-value
#> [1] 0.2425907
```

- **Advantages**:
  - Considers discrepancies across the entire distribution.
  - Robust to outliers due to equal weighting.
  - Simple to compute and interpret.
- **Limitations**:
  - Less sensitive to deviations in the tails compared to the Anderson-Darling test.
  - May be less powerful than the Kolmogorov-Smirnov test in detecting central shifts.

3.3.2.2.4 Jarque-Bera Test The **Jarque-Bera (JB) test** (Bera and Jarque, 1981) is a goodness-of-fit test used to check whether a dataset follows a normal distribution. It is based on the **skewness** and **kurtosis** of the data, which measure the asymmetry and the "tailedness" of the distribution, respectively.

The Jarque-Bera test statistic is defined as:

$$JB = \frac{n}{6}\left(S^2 + \frac{(K-3)^2}{4}\right),$$

where:

- n is the sample size,
- S is the sample skewness,
- K is the sample kurtosis.

Skewness (S) is calculated as:

$$S = \frac{\hat{\mu}_3}{\hat{\sigma}^3} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right)^{3/2}},$$

where:

- $\hat{\mu}_3$ is the third central moment,
- $\hat{\sigma}$ is the standard deviation,
- $\bar{x}$ is the sample mean.

Kurtosis (K) is calculated as:

$$K = \frac{\hat{\mu}_4}{\hat{\sigma}^4} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right)^{2}},$$

where:

- $\hat{\mu}_4$ is the fourth central moment.

Hypothesis

- **Null hypothesis (H_0)**: The data follow a normal distribution, implying:
  - Skewness $S = 0$,
  - Excess kurtosis $K - 3 = 0$.
- **Alternative hypothesis (H_1)**: The data do not follow a normal distribution.

Distribution of the JB Statistic

Under the null hypothesis, the Jarque-Bera statistic asymptotically follows a chi-squared distribution with 2 degrees of freedom:

$$JB \sim \chi^2_2.$$

Properties

1. **Sensitivity**:
   - Skewness (S) captures asymmetry in the data.
   - Kurtosis (K) measures how heavy-tailed or light-tailed the distribution is compared to a normal distribution.
2. **Limitations**:

- The test is sensitive to large sample sizes; even small deviations from normality may result in rejection of H_0.
- Assumes that the data are independently and identically distributed.

Applications

- Testing normality in regression residuals.
- Validating distributional assumptions in econometrics and time series analysis.

The Jarque-Bera test can be performed in R using the `tseries` package:

```
library(tseries)

# Generate a sample dataset
data <- rnorm(100)  # Normally distributed data

# Perform the Jarque-Bera test
jarque.bera.test(data)
#>
#>  Jarque Bera Test
#>
#> data:  data
#> X-squared = 1.6324, df = 2, p-value = 0.4421
```

3.4 Bivariate Statistics

Bivariate statistics involve the analysis of relationships between two variables. Understanding these relationships can provide insights into patterns, associations, or (suggestive of) causal connections. Below, we explore the correlation between different types of variables:

- Two Continuous **Variables**
- Two Discrete **Variables**
- Categorical and Continuous **Variables**

Before delving into the analysis, it is critical to consider the following:

1. **Is the relationship linear or non-linear?**
   - Linear relationships can be modeled with simpler statistical methods such as Pearson's correlation, while non-linear relationships may require alternative approaches, such as Spearman's rank correlation or regression with transformations.
2. **If the variable is continuous, is it normal and homoskedastic?**
   - For parametric methods like Pearson's correlation, assumptions such as normality and homoskedasticity (equal variance) must be met. When these assumptions fail, non-parametric methods like Spearman's correlation or robust alternatives are preferred.

3. **How big is your dataset?**
   - Large datasets can reveal subtle patterns but may lead to statistically significant results that are not practically meaningful. For smaller datasets, careful selection of statistical methods is essential to ensure reliability and validity.

	Categorical	Continuous
Categorical	Chi-squared Test	
	Phi Coefficient	
	Cramer's V	
	Tschuprow's T	
	Spearman's Rank Correlation	
	Kendall's Tau	
	Gamma Statistic	
	Freeman's Theta	
	Epsilon-squared	
	Goodman Kruskal's Gamma	
	Somers' D	
	Kendall's Tau-b	
	Yule's Q and Y	
	Tetrachoric Correlation	
	Polychoric Correlation	
Continuous	Point-Biserial Correlation	Pearson Correlation
	Logistic Regression	Spearman Correlation

3.4.1 Two Continuous

```r
set.seed(1)
n = 100 # (sample size)

data = data.frame(A = sample(1:20, replace = TRUE, size = n),
                  B = sample(1:30, replace = TRUE, size = n))
```

3.4.1.1 Pearson Correlation

Pearson correlation quantifies the strength and direction of a **linear relationship** between two continuous variables.

Formula:

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \cdot \sum(y_i - \bar{y})^2}}$$

where

- x_i, y_i: Individual data points of variables X and Y.
- $\bar{x}, \bar{y}$: Means of X and Y.

Assumptions:

1. The relationship between variables is **linear**.
2. Variables are **normally distributed**.
3. Data exhibits **homoscedasticity** (equal variance of Y for all values of X).

Use Case:

- Use when the relationship is expected to be linear, and assumptions of normality and homoscedasticity are met.

Interpretation:

- $r = +1$: Perfect positive linear relationship.
- $r = -1$: Perfect negative linear relationship.
- $r = 0$: No linear relationship.

```
# Pearson correlation
pearson_corr <- stats::cor(data$A, data$B, method = "pearson")
cat("Pearson Correlation (r):", pearson_corr, "\n")
#> Pearson Correlation (r): 0.02394939
```

3.4.1.2 Spearman Correlation

Spearman correlation measures the strength of a **monotonic relationship** between two variables. It ranks the data and calculates correlation based on ranks.

Formula:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

where

- d_i: Difference between the ranks of x_i and y_i.
- n: Number of paired observations.

Assumptions:

1. Relationship must be **monotonic**, not necessarily linear.
2. No assumptions about the distribution of variables.

Use Case:

- Use when data is ordinal or when normality and linearity assumptions are violated.

Interpretation:

- $\rho = +1$: Perfect positive monotonic relationship.
- $\rho = -1$: Perfect negative monotonic relationship.
- $\rho = 0$: No monotonic relationship.

```
# Spearman correlation
spearman_corr <- stats::cor(data$A, data$B, method = "spearman")
cat("Spearman Correlation (rho):", spearman_corr, "\n")
#> Spearman Correlation (rho): 0.02304636
```

3.4.1.3 Kendall's Tau Correlation

Kendall's Tau measures the strength of a **monotonic relationship** by comparing concordant and discordant pairs.

Formula:

$$\tau = \frac{(C - D)}{\binom{n}{2}}$$

where

- C: Number of concordant pairs (where ranks of X and Y increase or decrease together).
- D: Number of discordant pairs (where one rank increases while the other decreases).
- $\binom{n}{2}$: Total number of possible pairs.

Assumptions:

1. No specific assumptions about the data distribution.
2. Measures monotonic relationships.

Use Case:

- Preferred for small datasets or when data contains outliers.

Interpretation:

- $\tau = +1$: Perfect positive monotonic relationship.
- $\tau = -1$: Perfect negative monotonic relationship.
- $\tau = 0$: No monotonic relationship.

```
# Kendall's Tau correlation
kendall_corr <- stats::cor(data$A, data$B, method = "kendall")
```

```r
cat("Kendall's Tau Correlation (tau):", kendall_corr, "\n")
#> Kendall's Tau Correlation (tau): 0.02171284
```

3.4.1.4 Distance Correlation

Distance Correlation measures both **linear and non-linear relationships** between variables. It does not require monotonicity or linearity.

Formula:

$$dCor = \frac{dCov(X, Y)}{\sqrt{dVar(X) \cdot dVar(Y)}}$$

where

- $dCov$: Distance covariance between X and Y.
- $dVar$: Distance variances of X and Y.

Assumptions:

- No specific assumptions about the relationship (linear, monotonic, or otherwise).

Use Case:

- Use for complex relationships, including non-linear patterns.

Interpretation:

- $dCor = 0$: No association.
- $dCor = 1$: Perfect association.

```r
# Distance correlation
distance_corr <- energy::dcor(data$A, data$B)
cat("Distance Correlation (dCor):", distance_corr, "\n")
#> Distance Correlation (dCor): 0.1008934
```

3.4.1.5 Summary Table of Correlation Methods

Method	Formula/Approach	Detects Relationship Type	Assumptions	Sensitivity to Outliers	Use Case
Pearson	Linear covariance	Linear	Normality, homoscedasticity	High	Linear relationships.

Method	Formula/Approach	Detects Relationship Type	Assumptions	Sensitivity to Outliers	Use Case
Spearman	Ranks and monotonicity formula	Monotonic	None	Moderate	Monotonic, non-linear data.
Kendall's Tau	Concordance/discordance ratio	Monotonic	None	Low	Small datasets, robust to outliers.
Distance Correlation	Distance-based variance	Linear and non-linear	None	Low	Complex, non-linear relationships.

3.4.2 Categorical and Continuous

Analyzing the relationship between a **categorical variable** (binary or multi-class) and a **continuous variable** requires specialized techniques. These methods assess whether the categorical variable significantly influences the continuous variable or vice versa.

We focus on the following methods:

1. Point-Biserial Correlation
2. Logistic Regression
3. [Analysis of Variance (ANOVA)]
4. T-test

3.4.2.1 Point-Biserial Correlation

The **Point-Biserial Correlation** is a special case of the Pearson correlation used to assess the relationship between a **binary categorical variable** (coded as 0 and 1) and a continuous variable. It measures the strength and direction of the linear relationship.

Formula:

$$r_{pb} = \frac{\bar{Y_1} - \bar{Y_0}}{s_Y} \sqrt{\frac{n_1 n_0}{n^2}}$$

where

- $\bar{Y_1}$, $\bar{Y_0}$: Mean of the continuous variable for the groups coded as 1 and 0, respectively.
- s_Y: Standard deviation of the continuous variable.

- n_1, n_0: Number of observations in each group (1 and 0).
- n: Total number of observations.

Key Properties:

- **Range**: -1 to 1.
  - $r_{pb} = +1$: Perfect positive correlation.
  - $r_{pb} = -1$: Perfect negative correlation.
  - $r_{pb} = 0$: No linear relationship.
- A positive r_{pb} indicates higher values of the continuous variable are associated with the 1 group, while a negative r_{pb} indicates the opposite.

Assumptions:

1. The binary variable is **truly dichotomous** (e.g., male/female, success/failure).
2. The continuous variable is approximately **normally distributed**.
3. Homogeneity of variance across the two groups (not strictly required but recommended).

Use Case:

- To evaluate the linear relationship between a binary categorical variable and a continuous variable.

```
library(ltm)
# Point-Biserial Correlation
biserial_corr <- ltm::biserial.cor(
  c(12.5, 15.3, 10.7, 18.1, 11.2, 16.8, 13.4, 14.9),
  c(0, 1, 0, 1, 0, 1, 0, 1),
  use = "all.obs",
  level = 2
)
cat("Point-Biserial Correlation:", biserial_corr, "\n")
#> Point-Biserial Correlation: 0.8792835
```

3.4.2.2 Logistic Regression

Logistic Regression models the relationship between a **binary categorical variable** (dependent variable) and one or more independent variables (which may include continuous variables). It predicts the probability of the binary outcome (e.g., success/failure, yes/no).

Refer to 3.4.2.2 for more detail.

Formula:

The logistic regression model is represented as:

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X$$

where

- p: Probability of the outcome being 1.
- β_0: Intercept.
- β_1: Coefficient for the continuous variable X.
- $\text{logit}(p)$: Log-odds of the probability.

Key Features:

- **Output**: Odds ratio or probability of the binary outcome.
- Can include multiple predictors (continuous and categorical).
- Non-linear transformation ensures predictions are probabilities between 0 and 1.

Assumptions:

1. The dependent variable is **binary**.
2. Observations are **independent**.
3. There is a **linear relationship** between the logit of the dependent variable and the independent variable.
4. No **multicollinearity** between predictors.

Use Case:

- To predict the likelihood of a binary outcome based on a continuous predictor (e.g., probability of success given test scores).

```r
# Simulated data
set.seed(123)
x <- rnorm(100, mean = 50, sd = 10)  # Continuous predictor
y <- ifelse(x > 55, 1, 0)  # Binary outcome based on threshold

# Logistic Regression
logistic_model <- glm(y ~ x, family = binomial)
summary(logistic_model)
#>
#> Call:
#> glm(formula = y ~ x, family = binomial)
#>
#> Deviance Residuals:
#>        Min          1Q      Median          3Q         Max
#> -2.770e-04  -2.100e-08  -2.100e-08   2.100e-08   2.548e-04
```

```
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  -3749.9   495083.0  -0.008    0.994
#> x               67.9     8966.6   0.008    0.994
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 1.2217e+02  on 99  degrees of freedom
#> Residual deviance: 1.4317e-07  on 98  degrees of freedom
#> AIC: 4
#>
#> Number of Fisher Scoring iterations: 25

# Predicted probabilities
predicted_probs <- predict(logistic_model, type = "response")
print(head(predicted_probs))
#>         1         2         3         4         5         6
#> -735.6466 -511.3844  703.2134 -307.2281 -267.3187  809.3747
```

```
# Visualize logistic regression curve
library(ggplot2)
data <- data.frame(x = x, y = y, predicted = predicted_probs)

ggplot(data, aes(x = x, y = predicted)) +
    geom_point(aes(y = y), color = "red", alpha = 0.5) +
    geom_line(color = "blue") +
    labs(title = "Logistic Regression: Continuous vs Binary",
         x = "Continuous Predictor", y = "Predicted Probability")
```

3.4.2.3 Summary Table of Methods (Between Categorical and Continuous)

Method	Type of Variable Relationship	Key Assumptions	Use Case
Point-Biserial Correlation	Binary Categorical vs Continuous	Linear, normality (continuous)	Assess linear association.
Logistic Regression	Continuous → Binary Categorical	Logit-linear relationship	Predict probability of binary outcome.
ANOVA	Multi-level Categorical vs Continuous	Normality, homogeneity of variance	Compare means across groups.
T-Test	Binary Categorical vs Continuous	Normality, equal variance	Compare means between two groups.

3.4.3 Two Discrete

When analyzing the relationship between two **discrete variables** (categorical or ordinal), various methods are available to quantify the degree of association or similarity. These methods can broadly be classified into:

1. Distance Metrics
2. Statistical Metrics

3.4.3.1 Distance Metrics

Distance metrics measure the **dissimilarity** between two discrete variables and are often used as a proxy for correlation in specific applications like clustering or machine learning.

3.4.3.1.1 Euclidean Distance

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- Measures the straight-line distance between two variables in Euclidean space.
- Sensitive to scaling; variables should be normalized for meaningful comparisons.

3.4.3.1.2 Manhattan Distance

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

- Measures distance by summing the absolute differences along each dimension.
- Also called **L1 norm**; often used in grid-based problems.

3.4.3.1.3 Chebyshev Distance

$$d(x, y) = \max_{i=1}^{n} |x_i - y_i|$$

- Measures the maximum single-step distance along any dimension.
- Useful in discrete, grid-based problems (e.g., chess moves).

3.4.3.1.4 Minkowski Distance

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

- Generalized distance metric. Special cases include:
  - $p = 1$: Manhattan Distance.
  - $p = 2$: Euclidean Distance.
  - $p \to \infty$: Chebyshev Distance.

3.4.3.1.5 Canberra Distance

$$d(x, y) = \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|}$$

- Emphasizes proportional differences, making it sensitive to smaller values.

3.4.3.1.6 Hamming Distance

$$d(x, y) = \sum_{i=1}^{n} I(x_i \neq y_i)$$

- Counts the number of differing positions between two sequences.
- Widely used in text similarity and binary data.

3.4.3.1.7 Cosine Similarity and Distance

$$\text{Cosine Similarity} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \cdot \sqrt{\sum_{i=1}^{n} y_i^2}}$$

$$\text{Cosine Distance} = 1 - \text{Cosine Similarity}$$

- Measures the angle between two vectors in a high-dimensional space.
- Often used in text and document similarity.

3.4.3.1.8 Sum of Absolute Differences

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

- Equivalent to Manhattan Distance but without coordinate context.

3.4.3.1.9 Sum of Squared Differences

$$d(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2$$

- Equivalent to squared Euclidean Distance.

3.4.3.1.10 Mean Absolute Error

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |x_i - y_i|$$

- Measures average absolute differences.

```r
# Example data
x <- c(1, 2, 3, 4, 5)
y <- c(2, 3, 4, 5, 6)

# Compute distances
euclidean <- sqrt(sum((x - y)^2))
manhattan <- sum(abs(x - y))
chebyshev <- max(abs(x - y))
hamming <- sum(x != y)
cosine_similarity <- sum(x * y) / (sqrt(sum(x^2)) * sqrt(sum(y^2)))
cosine_distance <- 1 - cosine_similarity

# Display results
cat("Euclidean Distance:", euclidean, "\n")
#> Euclidean Distance: 2.236068
cat("Manhattan Distance:", manhattan, "\n")
#> Manhattan Distance: 5
cat("Chebyshev Distance:", chebyshev, "\n")
#> Chebyshev Distance: 1
cat("Hamming Distance:", hamming, "\n")
#> Hamming Distance: 5
cat("Cosine Distance:", cosine_distance, "\n")
#> Cosine Distance: 0.005063324
```

3.4.3.2 Statistical Metrics

3.4.3.2.1 Chi-squared Test The **Chi-Squared Test** evaluates whether two categorical variables are **independent** by comparing observed and expected frequencies in a contingency table.

Formula:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

where

- O_i: Observed frequency in each cell of the table.
- E_i: Expected frequency under the assumption of independence.

Steps:

1. Construct a contingency table with observed counts.
2. Compute expected frequencies: $E_{ij} = \frac{\text{Row Total}_i \cdot \text{Column Total}_j}{\text{Grand Total}}$
3. Apply the Chi-squared formula.

4. Compare χ^2 with a critical value from the Chi-squared distribution.

Assumptions:

1. Observations are **independent**.
2. Expected frequencies should be ≥ 5 in at least 80% of the cells.

Use Case:

- Tests for **independence** between two nominal variables.

```
# Example data
dt <- matrix(c(15, 25, 20, 40), nrow = 2)
rownames(dt) <- c("Group A", "Group B")
colnames(dt) <- c("Category 1", "Category 2")

# Perform Chi-Squared Test
chi_sq_test <- chisq.test(dt)
print(chi_sq_test)
#>
#>  Pearson's Chi-squared test with Yates' continuity correction
#>
#> data:  dt
#> X-squared = 0.045788, df = 1, p-value = 0.8306
```

3.4.3.2.2 Phi Coefficient The **Phi Coefficient** is a measure of association between two **binary variables**, derived from the Chi-squared statistic.

Formula:

$$\phi = \frac{\chi^2}{n}$$

where

- n: Total sample size.

Interpretation:

- $\phi = 0$: No association.
- $\phi = +1$: Perfect positive association.
- $\phi = -1$: Perfect negative association.

Use Case:

- Suitable for **2x2 contingency tables**.
- 2 binary

```r
library(psych)

# Compute Phi Coefficient
phi_coeff <- phi(dt)
cat("Phi Coefficient:", phi_coeff, "\n")
#> Phi Coefficient: 0.04
```

3.4.3.2.3 Cramer's V **Cramer's V** generalizes the Phi coefficient to handle contingency tables with more than two rows or columns.

Formula:

$$V = \sqrt{\frac{\chi^2/n}{\min(r-1, c-1)}}$$

where

- r: Number of rows.
- c: Number of columns.

Assumptions:

1. Variables are nominal.
2. Suitable for **larger contingency tables**.

Use Case:

- Measures the **strength of association** between nominal variables with no natural order.

```r
library(lsr)

# Simulate data
set.seed(1)
data <- data.frame(
  A = sample(1:5, replace = TRUE, size = 100),  # Nominal variable
  B = sample(1:6, replace = TRUE, size = 100)  # Nominal variable
)

# Compute Cramer's V
cramers_v <- cramersV(data$A, data$B)
cat("Cramer's V:", cramers_v, "\n")
#> Cramer's V: 0.1944616
```

Alternatively,

- `ncchisq` noncentral Chi-square

- `nchisqadj` Adjusted noncentral Chi-square
- `fisher` Fisher Z transformation
- `fisheradj` bias correction Fisher z transformation

```
DescTools::CramerV(data, conf.level = 0.95,method = "ncchisqadj")
#>  Cramer V    lwr.ci    upr.ci
#> 0.3472325 0.3929964 0.4033053
```

3.4.3.2.4 Adjusted Cramer's V Adjusted versions of Cramer's V correct for bias, especially in small samples.

Adjusted formulas account for non-central Chi-squared or bias correction. Examples include:

- **Non-central Chi-squared**: $V_{adj} = \sqrt{\frac{\chi^2_{nc}/n}{\min(r-1,c-1)}}$
- **Bias Correction**: $V_{adj} = V - \text{Bias Term}$

```
library(DescTools)

# Compute Adjusted Cramer's V
cramers_v_adj <- CramerV(data, conf.level = 0.95, method = "ncchisqadj")
cramers_v_adj
#>  Cramer V    lwr.ci    upr.ci
#> 0.3472325 0.3929964 0.4033053
```

3.4.3.2.5 Tschuprow's T **Tschuprow's T** is a symmetric measure of association for nominal variables. It differs from Cramer's V by considering the product of rows and columns, making it less sensitive to asymmetrical tables.

Formula:

$$T = \sqrt{\frac{\chi^2/n}{\sqrt{(r-1)(c-1)}}}$$

Assumptions:

- Applicable to nominal variables.
- Suitable for contingency tables with unequal dimensions.

Use Case:

- Preferred when table dimensions are **highly unequal**.

```
# Compute Tschuprow's T
tschuprow_t <- DescTools::TschuprowT(data$A, data$B)
```

```r
cat("Tschuprow's T:", tschuprow_t, "\n")
#> Tschuprow's T: 0.1839104
```

3.4.3.2.6 Ordinal Association (Rank correlation) When at least one variable is **ordinal**, rank-based methods are the most appropriate as they respect the **order of the categories**. These methods are often used when relationships are monotonic (increasing or decreasing consistently) but not necessarily linear.

3.4.3.2.6.1 Spearman's Rank Correlation **Spearman's Rank Correlation** (ρ) measures the strength and direction of a **monotonic relationship** between two variables. It transforms the data into ranks and calculates Pearson correlation on the ranks.

Formula:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

where

- d_i: Difference between the ranks of the paired observations.
- n: Number of paired observations.

Assumptions:

1. Data must be **ordinal** or **continuous** but convertible to ranks.
2. Relationship is **monotonic**.

Use Case:

- Suitable for ordinal-ordinal or ordinal-continuous associations.

```r
# Simulating ordinal data
set.seed(123)
ordinal_x <- sample(1:5, 100, replace = TRUE)
ordinal_y <- sample(1:5, 100, replace = TRUE)

# Spearman's Correlation
spearman_corr <- cor(ordinal_x, ordinal_y, method = "spearman")
cat("Spearman's Correlation (rho):", spearman_corr, "\n")
#> Spearman's Correlation (rho): 0.08731195
```

3.4.3.2.6.2 Kendall's Tau **Kendall's Tau** (τ) measures the strength of a **monotonic relationship** by comparing concordant and discordant pairs.

Formula:

$$\tau = \frac{C - D}{C + D}$$

where

- C: Number of concordant pairs (ranks increase together).
- D: Number of discordant pairs (one rank increases while the other decreases).

Variants:

1. **Kendall's Tau-a**: For data with no ties.
2. **Kendall's Tau-b**: Adjusted for ties in ranks.
3. **Kendall's Tau-c**: Adjusted for ties in large tables.

Use Case:

- Ideal for small datasets or when ties are present.

```
# Kendall's Tau
kendall_corr <- cor(ordinal_x, ordinal_y, method = "kendall")
cat("Kendall's Tau (tau):", kendall_corr, "\n")
#> Kendall's Tau (tau): 0.06795076
```

3.4.3.2.6.3 Gamma Statistic The **Gamma Statistic** measures the strength of association between two ordinal variables by focusing on **concordant** and **discordant** pairs, ignoring ties.

Formula:

$$\gamma = \frac{C - D}{C + D}$$

Use Case:

- Works well when there are **many ties** in the data.

```
library(vcd)

# Simulating ordinal data
cont_table <- table(ordinal_x, ordinal_y)

# Gamma Statistic
gamma_stat <- assocstats(cont_table)$gamma
cat("Gamma Statistic:", gamma_stat, "\n")
#> Gamma Statistic:
```

3.4.3.2.6.4 Freeman's Theta **Freeman's Theta** measures the association between an **ordinal variable** and a **nominal variable**. It quantifies how well the grouping in the nominal variable explains the ordering in the ordinal variable.

Use Case:

- Useful when analyzing relationships between ordinal predictors and nominal responses (or vice versa).

```
rcompanion::freemanTheta(ordinal_x, ordinal_y)
#> Freeman.theta
#>         0.094
```

3.4.3.2.6.5 Epsilon-squared **Epsilon-Squared** (ϵ^2) measures the proportion of variance in the **ordinal variable** explained by a **nominal variable**. It is conceptually similar to the coefficient of determination (R^2) in linear regression but adapted for ordinal-nominal relationships.

Formula:

$$\epsilon^2 = \frac{\text{variance explained by group differences}}{\text{total variance}}$$

where

- The numerator represents the variance between ordinal categories due to differences in nominal groups.
- The denominator is the total variance in the ordinal variable.

Use Case:

- Quantifies the effect size when analyzing how well a nominal variable explains an ordinal variable.

```
set.seed(123)
ordinal_x <- sample(1:5, 100, replace = TRUE)  # Ordinal variable
nominal_y <- sample(1:3, 100, replace = TRUE) # Nominal variable

# Compute Epsilon-Squared
library(rcompanion)
epsilon_squared <- rcompanion::epsilonSquared(ordinal_x,  nominal_y)
print(epsilon_squared)
#> epsilon.squared
#>         0.00446
```

3.4.3.2.6.6 Goodman-Kruskal's Gamma **Goodman-Kruskal's Gamma** measures the strength of association between two **ordinal variables**.

It is a rank-based measure, focusing only on **concordant** and **discordant** pairs while ignoring ties.

Formula:

$$\gamma = \frac{C - D}{C + D}$$

where

- C: Number of concordant pairs (where ranks move in the same direction).
- D: Number of discordant pairs (where ranks move in opposite directions).

Use Case:

- Suitable for ordinal variables with many ties.

```
n = 100 # (sample size)
set.seed(1)
dt = table(data.frame(
    A = sample(1:4, replace = TRUE, size = n), # ordinal
    B = sample(1:3, replace = TRUE, size = n)  # ordinal
))
dt
#>    B
#> A    1  2  3
#>   1  7 11  9
#>   2 11  6 14
#>   3  7 11  4
#>   4  6  4 10
```

```
# Compute Goodman-Kruskal's Gamma
library(DescTools)
goodman_kruskal_gamma <- GoodmanKruskalGamma(dt, conf.level = 0.95)
cat("Goodman-Kruskal's Gamma:", goodman_kruskal_gamma, "\n")
#> Goodman-Kruskal's Gamma: 0.006781013 -0.2290321 0.2425941
```

3.4.3.2.6.7 Somers' D **Somers' D** (also called Somers' Delta) extends Kendall's Tau by focusing on **asymmetric relationships**, where one variable is a predictor and the other is a response.

Formula:

$$D_{XY} = \frac{C - D}{C + D + T_Y}$$

where

- T_Y: Tied pairs in the dependent variable.

Use Case:

- Appropriate when there is a clear predictor-response relationship between two ordinal variables.

```
# Compute Somers' D
somers_d <- SomersDelta(dt, conf.level = 0.95)
somers_d
#>       somers       lwr.ci       upr.ci
#>  0.005115859 -0.172800185  0.183031903
```

3.4.3.2.6.8 Kendall's Tau-b **Kendall's Tau-b** is an extension of Kendall's Tau that accounts for **ties** in the data.

Formula:

$$\tau_b = \frac{C - D}{\sqrt{(C + D + T_X)(C + D + T_Y)}}$$

where

- T_X, T_Y: Tied pairs in each variable.

Use Case:

- Use when ordinal data contains ties.

```
# Compute Kendall's Tau-b
kendalls_tau_b <- KendallTauB(dt, conf.level = 0.95)
kendalls_tau_b
#>        tau_b       lwr.ci       upr.ci
#>  0.004839732 -0.163472443  0.173151906
```

3.4.3.2.6.9 Yule's Q and Y **Yule's Q** and **Yule's Y** are specialized measures for **2x2 contingency tables**. They are simplified versions of Goodman-Kruskal's Gamma, designed for binary ordinal variables.

Use Case:

- Ideal for binary ordinal variables in a 2x2 table.

Special version (2×2) of the Goodman Kruskal's Gamma coefficient.

	Variable 1	
Variable 2	a	b
	c	d

$$\text{Yule's Q} = \frac{ad - bc}{ad + bc}$$

We typically use Yule's Q in practice while Yule's Y has the following relationship with Q.

$$\text{Yule's Y} = \frac{\sqrt{ad} - \sqrt{bc}}{\sqrt{ad} + \sqrt{bc}}$$

$$Q = \frac{2Y}{1 + Y^2}$$

$$Y = \frac{1 = \sqrt{1 - Q^2}}{Q}$$

```
# Create 2x2 table
dt_binary <- table(data.frame(
  A = sample(c(0, 1), replace = TRUE, size = n),
  B = sample(c(0, 1), replace = TRUE, size = n)
))

# Compute Yule's Q
yules_q <- YuleQ(dt_binary)
yules_q
#> [1] -0.07667474
```

3.4.3.2.6.10 Tetrachoric Correlation **Tetrachoric Correlation** measures the association between two **binary variables** by assuming they represent thresholds of underlying continuous normal distributions. It is a special case of Polychoric Correlation when both variables are binary

```
# Simulate binary data
library(psych)
data_binary <- data.frame(
  A = sample(c(0, 1), replace = TRUE, size = n),
  B = sample(c(0, 1), replace = TRUE, size = n)
)

# Compute Tetrachoric Correlation
tetrachoric_corr <- tetrachoric(data_binary)
print(tetrachoric_corr)
#> Call: tetrachoric(x = data_binary)
#> tetrachoric correlation
#>   A    B
```

```
#> A 1.00
#> B 0.31 1.00
#>
#>  with tau of
#>      A      B
#>  0.126 -0.025
```

3.4.3.2.6.11 Polychoric Correlation **Polychoric Correlation** measures the association between **ordinal variables** by assuming they are discretized versions of latent, normally distributed continuous variables.

Assumptions:

- The ordinal variables represent categories of an underlying normal distribution.

Use Case:

- Suitable for ordinal variables with a natural order.

```r
# Simulate ordinal data
library(polycor)
data_ordinal <- data.frame(
  A = sample(1:4, replace = TRUE, size = n),
  B = sample(1:6, replace = TRUE, size = n)
)

# Compute Polychoric Correlation
polychoric_corr <- polychor(data_ordinal$A, data_ordinal$B)
cat("Polychoric Correlation:", polychoric_corr, "\n")
#> Polychoric Correlation: 0.1908334
```

Metric	Variable Types	Use Case
Spearman's Correlation	Ordinal vs. Ordinal	Non-linear, monotonic relationships.
Kendall's Tau	Ordinal vs. Ordinal	Non-linear, monotonic relationships with ties.
Gamma Statistic	Ordinal vs. Ordinal	Handles data with many ties effectively.
Freeman's Theta	Ordinal vs. Nominal	Mixed data types (ordinal and nominal).
Epsilon-Squared	Ordinal vs. Nominal	Variance explained by nominal groups.
Goodman-Kruskal's Gamma	Ordinal vs. Ordinal	Strong association; ignores ties.

Metric	Variable Types	Use Case
Somers' D	Ordinal Predictor and Response	Asymmetric association.
Kendall's Tau-b	Ordinal vs. Ordinal	Adjusts for ties in data.
Yule's Q	Binary Ordinal vs. Binary Ordinal	Special case for 2x2 tables.
Tetrachoric Correlation	Binary vs. Binary	Binary ordinal variables.
Polychoric Correlation	Ordinal vs. Ordinal	Continuous latent structure.

3.4.4 General Approach to Bivariate Statistics

```
library(tidyverse)

data("mtcars")
df = mtcars %>%
    dplyr::select(cyl, vs, carb)


df_factor = df %>%
    dplyr::mutate(
        cyl = factor(cyl),
        vs = factor(vs),
        carb = factor(carb)
    )
# summary(df)
str(df)
#> 'data.frame':    32 obs. of  3 variables:
#>  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
#>  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
#>  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
str(df_factor)
#> 'data.frame':    32 obs. of  3 variables:
#>  $ cyl : Factor w/ 3 levels "4","6","8": 2 2 1 2 3 2 3 1 1 2 ...
#>  $ vs  : Factor w/ 2 levels "0","1": 1 1 2 2 1 2 1 2 2 2 ...
#>  $ carb: Factor w/ 6 levels "1","2","3","4",..: 4 4 1 1 2 1 4 2 2 4 ...
```

Get the correlation table for continuous variables only

```
cor(df)
#>             cyl         vs       carb
#> cyl   1.0000000 -0.8108118  0.5269883
#> vs   -0.8108118  1.0000000 -0.5696071
#> carb  0.5269883 -0.5696071  1.0000000
```

	cyl	vs	carb
cyl	1	.	.
vs	−0.81	1	.
carb	0.53	−0.57	1

```
# only complete obs
# cor(df, use = "complete.obs")
```

Alternatively, you can also have the

```
Hmisc::rcorr(as.matrix(df), type = "pearson")
#>        cyl    vs  carb
#> cyl   1.00 -0.81  0.53
#> vs   -0.81  1.00 -0.57
#> carb  0.53 -0.57  1.00
#>
#> n= 32
#>
#>
#> P
#>      cyl    vs     carb
#> cyl         0.0000 0.0019
#> vs   0.0000        0.0007
#> carb 0.0019 0.0007
```

```
modelsummary::datasummary_correlation(df)
```

```
ggcorrplot::ggcorrplot(cor(df))
```

Comparing correlations between different types of variables (e.g., continuous vs. categorical) poses unique challenges. One key issue is ensuring that methods are appropriate for the nature of the variables being analyzed. Another challenge lies in detecting non-linear relationships, as traditional correlation measures, such as Pearson's correlation coefficient, are designed to assess linear associations.

To address these challenges, a potential solution is to utilize **mutual information** from information theory. Mutual information quantifies how much knowing one variable reduces the uncertainty of another, providing a more general measure of association that accommodates both linear and non-linear relationships.

3.4.4.1 Approximating Mutual Information

We can approximate mutual information using the following relationship:

$$\downarrow \text{Prediction Error} \approx \downarrow \text{Uncertainty} \approx \uparrow \text{Association Strength}$$

This principle underpins the X2Y metric, which is implemented through the following steps:

1. **Predict** y without x (baseline model):
    - If y is continuous, predict the mean of y.
    - If y is categorical, predict the mode of y.

2. **Predict** y with x using a model (e.g., linear regression, random forest, etc.).
3. **Calculate the difference in prediction error** between steps 1 and 2. This difference reflects the reduction in uncertainty about y when x is included, serving as a measure of association strength.

3.4.4.2 Generalizing Across Variable Types

To construct a comprehensive framework that handles different variable combinations, such as:

- Continuous vs. continuous
- Categorical vs. continuous
- Continuous vs. categorical
- Categorical vs. categorical

a flexible modeling approach is required. **Classification and Regression Trees (CART)** are particularly well-suited for this purpose, as they can accommodate both continuous and categorical predictors and outcomes. However, other models, such as random forests or generalized additive models (GAMs), may also be employed.

3.4.4.3 Limitations of the Approach

Despite its strengths, this approach has some limitations:

1. **Asymmetry:**
   The measure is not symmetric, meaning $(x, y) \neq (y, x)$.
2. **Comparability:**
   Different variable pairs may yield metrics that are not directly comparable. For instance, continuous outcomes often use metrics like Mean Absolute Error (MAE), while categorical outcomes use measures like misclassification error.

These limitations should be considered when interpreting results, especially in multi-variable or mixed-data contexts.

```
library(ppsr)
library(tidyverse)

iris <- iris %>%
  dplyr::select(1:3)

# ppsr::score_df(iris) # if you want a dataframe
ppsr::score_matrix(iris,
                   do_parallel = TRUE,
                   n_cores = parallel::detectCores() / 2)
#>              Sepal.Length Sepal.Width Petal.Length
```

```
#> Sepal.Length   1.00000000  0.04632352    0.5491398
#> Sepal.Width    0.06790301  1.00000000    0.2376991
#> Petal.Length   0.61608360  0.24263851    1.0000000

# if you want a similar correlation matrix
ppsr::score_matrix(df,
                   do_parallel = TRUE,
                   n_cores = parallel::detectCores() / 2)
#>              cyl        vs      carb
#> cyl  1.00000000 0.3982789 0.2092533
#> vs   0.02514286 1.0000000 0.2000000
#> carb 0.30798148 0.2537309 1.0000000
```

```
corrplot::corrplot(cor(df))
```

Alternatively,

```
PerformanceAnalytics::chart.Correlation(df, histogram = T, pch = 19)
```

```
heatmap(as.matrix(df))
```

More general form,

```
ppsr::visualize_pps(
    df = iris,
    do_parallel = TRUE,
    n_cores = parallel::detectCores() / 2
)
```

```
ppsr::visualize_correlations(
    df = iris
)
```

Both heat map and correlation at the same time

```
ppsr::visualize_both(
    df = iris,
    do_parallel = TRUE,
```

```
    n_cores = parallel::detectCores() / 2
)
```

More elaboration with `ggplot2`

```
ppsr::visualize_pps(
    df = iris,
    color_value_high = 'red',
    color_value_low = 'yellow',
    color_text = 'black'
) +
    ggplot2::theme_classic() +
    ggplot2::theme(plot.background =
                       ggplot2::element_rect(fill = "lightgrey")) +
    ggplot2::theme(title = ggplot2::element_text(size = 15)) +
    ggplot2::labs(
        title = 'Correlation aand Heatmap',
        subtitle = 'Subtitle',
        caption = 'Caption',
        x = 'More info'
    )
```

Correlation aand Heatmap
Subtitle
target
Sepal.Length
Sepal.Width
Petal.Length
1.00
0.05
0.55
0.07
1.00
0.24
0.62
0.24
1.00
Sepal.Length
Sepal.Width
Petal.Length
More info
Caption
pps
1.0
0.8
0.6
0.4
0.2
0.0

Chapter 4

Basic Statistical Inference

Statistical inference involves drawing conclusions about population parameters based on sample data. The two primary goals of inference are:

1. **Making inferences** about the true parameter value (β) based on our estimator or estimate:
   - This involves interpreting the sample-derived estimate to understand the population parameter.
   - Examples include estimating population means, variances, or proportions.
2. **Testing whether underlying assumptions hold true**, including:
   - Assumptions about the true population parameters (e.g., μ, σ^2).
   - Assumptions about random variables (e.g., independence, normality).
   - Assumptions about the model specification (e.g., linearity in regression).

Note: Statistical testing does not:

- Confirm with absolute certainty that a hypothesis is true or false.
- Interpret the magnitude of the estimated value in economic, practical, or business contexts without additional analysis.
  - **Statistical significance**: Refers to whether an observed effect is unlikely due to chance.
  - **Practical significance**: Focuses on the real-world importance of the effect.

Example:

- A marketing campaign increases sales by 0.5%, which is statistically significant ($p < 0.05$). However, in a small market, this may lack practical

significance.

Instead, inference provides a framework for making probabilistic statements about population parameters, given sample data.

4.1 Hypothesis Testing Framework

Hypothesis testing is one of the fundamental tools in statistics. It provides a formal procedure to test claims or assumptions (hypotheses) about population parameters using sample data. This process is essential in various fields, including business, medicine, and social sciences, as it helps answer questions like "Does a new marketing strategy improve sales?" or "Is there a significant difference in test scores between two teaching methods?"

The goal of hypothesis testing is to make decisions or draw conclusions about a population based on sample data. This is necessary because we rarely have access to the entire population. For example, if a company wants to determine whether a new advertising campaign increases sales, it might analyze data from a sample of stores rather than every store globally.

Key Steps in Hypothesis Testing

1. **Formulate Hypotheses**: Define the null and alternative hypotheses.
2. **Choose a Significance Level** (α): Determine the acceptable probability of making a Type I error.
3. **Select a Test Statistic**: Identify the appropriate statistical test based on the data and hypotheses.
4. **Define the Rejection Region**: Specify the range of values for which the null hypothesis will be rejected.
5. **Compute the Test Statistic**: Use sample data to calculate the test statistic.
6. **Make a Decision**: Compare the test statistic to the critical value or use the p-value to decide whether to reject or fail to reject the null hypothesis.

4.1.1 Null and Alternative Hypotheses

At the heart of hypothesis testing lies the formulation of two competing hypotheses:

1. **Null Hypothesis (H_0)**:
   - Represents the current state of knowledge, status quo, or no effect.
   - It is assumed true unless there is strong evidence against it.
   - Examples:
     - $H_0 : \mu_1 = \mu_2$ (no difference in means between two groups).

  - $H_0 : \beta = 0$ (a predictor variable has no effect in a regression model).
- Think of H_0 as the "default assumption."

2. **Alternative Hypothesis (H_a or H_1):**
   - Represents a claim that contradicts the null hypothesis.
   - It is what you are trying to prove or find evidence for.
   - Examples:
     - $H_a : \mu_1 \neq \mu_2$ (means of two groups are different).
     - $H_a : \beta \neq 0$ (a predictor variable has an effect).

4.1.2 Errors in Hypothesis Testing

Hypothesis testing involves decision-making under uncertainty, meaning there is always a risk of making errors. These errors are classified into two types:

1. **Type I Error** (α):
   - Occurs when the null hypothesis is rejected, even though it is true.
   - Example: Concluding that a medication is effective when it actually has no effect.
   - The probability of making a Type I error is denoted by α, called the **significance level** (commonly set at 0.05 or 5%).
2. **Type II Error** (β):
   - Occurs when the null hypothesis is not rejected, but the alternative hypothesis is true.
   - Example: Failing to detect that a medication is effective when it actually works.
   - The complement of β is called the **power** of the test $(1 - \beta)$, representing the probability of correctly rejecting the null hypothesis.

Analogy: The Legal System

To make this concept more intuitive, consider the analogy of a courtroom:

- **Null Hypothesis (H_0):** The defendant is innocent.
- **Alternative Hypothesis (H_a):** The defendant is guilty.
- **Type I Error:** Convicting an innocent person (false positive).
- **Type II Error:** Letting a guilty person go free (false negative).

Balancing α and β is critical in hypothesis testing, as reducing one often increases the other. For example, if you make it harder to reject H_0 (reducing α), you increase the chance of failing to detect a true effect (increasing β).

4.1.3 The Role of Distributions in Hypothesis Testing

Distributions play a fundamental role in hypothesis testing because they provide a mathematical model for understanding how a test statistic behaves under the null hypothesis (H_0). Without distributions, it would be impossible to determine whether the observed results are due to random chance or provide evidence to reject the null hypothesis.

4.1.3.1 Expected Outcomes

One of the key reasons distributions are so crucial is that they describe the range of values a test statistic is likely to take when H_0 is true. This helps us understand what is considered "normal" variation in the data due to random chance. For example:

- Imagine you are conducting a study to test whether a new marketing strategy increases the average monthly sales. Under the null hypothesis, you assume the new strategy has no effect, and the average sales remain unchanged.
- When you collect a sample and calculate the test statistic, you compare it to the expected distribution (e.g., the normal distribution for a z-test). This distribution shows the range of test statistic values that are likely to occur purely due to random fluctuations in the data, assuming H_0 is true.

By providing this baseline of what is "normal," distributions allow us to identify unusual results that may indicate the null hypothesis is false.

4.1.3.2 Critical Values and Rejection Regions

Distributions also help define critical values and rejection regions in hypothesis testing. Critical values are specific points on the distribution that mark the boundaries of the rejection region. The rejection region is the range of values for the test statistic that lead us to reject H_0.

The location of these critical values depends on:

- The **level of significance** (α), which is the probability of rejecting H_0 when it is true (a Type I error).
- The shape of the test statistic's distribution under H_0.

For example:

- In a one-tailed z-test with $\alpha = 0.05$, the critical value is approximately 1.645 for a standard normal distribution. If the calculated test statistic exceeds this value, we reject H_0 because such a result would be very unlikely under H_0.

Distributions help us visually and mathematically determine these critical points. By examining the distribution, we can see where the rejection region

lies and what the probability is of observing a value in that region by random chance alone.

4.1.3.3 P-values

The p-value, a central concept in hypothesis testing, is directly derived from the distribution of the test statistic under H_0. The p-value represents the probability of observing a test statistic as extreme as (or more extreme than) the one calculated, assuming H_0 is true.

The **p-value** quantifies the strength of evidence against H_0. It represents the probability of observing a test statistic as extreme as (or more extreme than) the one calculated, assuming H_0 is true.

- **Small p-value** ($< \alpha$): Strong evidence against H_0; reject H_0.
- **Large p-value** ($> \alpha$): Weak evidence against H_0; fail to reject H_0.

For example:

- Suppose you calculate a z-test statistic of 2.1 in a one-tailed test. Using the standard normal distribution, the p-value is the area under the curve to the right of $z = 2.1$. This area represents the likelihood of observing a result as extreme as $z = 2.1$ if H_0 is true.

- In this case, the p-value is approximately 0.0179. A small p-value (typically less than $\alpha = 0.05$) suggests that the observed result is unlikely under H_0 and provides evidence to reject the null hypothesis.

4.1.3.4 Why Does All This Matter?

To summarize, distributions are the backbone of hypothesis testing because they allow us to:

- Define what is expected under H_0 by modeling the behavior of the test statistic.

- Identify results that are unlikely to occur by random chance, which leads to the rejection of H_0.

- Calculate p-values to quantify the strength of evidence against H_0.

Distributions provide the framework for understanding the role of chance in statistical analysis. They are essential for determining expected outcomes, setting thresholds for decision-making (critical values and rejection regions), and calculating p-values. A solid grasp of distributions will greatly enhance your ability to interpret and conduct hypothesis tests, making it easier to draw meaningful conclusions from data.

4.1.4 The Test Statistic

The test statistic is a crucial component in hypothesis testing, as it quantifies how far the observed data deviates from what we would expect if the null hypothesis (H_0) were true. Essentially, it provides a standardized way to compare the observed outcomes against the expectations set by H_0, enabling us to assess whether the observed results are likely due to random chance or indicative of a significant effect.

The general formula for a test statistic is:

$$\text{Test Statistic} = \frac{\text{Observed Value} - \text{Expected Value under } H_0}{\text{Standard Error}}$$

Each component of this formula has an important role:

1. **Numerator:**
   - The numerator represents the difference between the actual data (observed value) and the hypothetical value (expected value) that is assumed under H_0.
   - This difference quantifies the extent of the deviation. A larger deviation suggests stronger evidence against H_0.
2. **Denominator:**
   - The denominator is the **standard error**, which measures the variability or spread of the data. It accounts for factors such as sample size and the inherent randomness of the data.
   - By dividing the numerator by the standard error, the test statistic is standardized, allowing comparisons across different studies, sample sizes, and distributions.

The test statistic plays a central role in determining whether to reject H_0. Once calculated, it is compared to a known distribution (e.g., standard normal distribution for z-tests or t-distribution for t-tests). This comparison allows us to evaluate the likelihood of observing such a test statistic under H_0:

- **If the test statistic is close to 0:** This indicates that the observed data is very close to what is expected under H_0. There is little evidence to suggest rejecting H_0.
- **If the test statistic is far from 0 (in the tails of the distribution):** This suggests that the observed data deviates significantly from the expectations under H_0. Such deviations may provide strong evidence against H_0.

4.1.4.1 Why Standardizing Matters

Standardizing the difference between the observed and expected values ensures that the test statistic is not biased by factors such as the scale of measurement or the size of the sample. For instance:

- A raw difference of 5 might be highly significant in one context but negligible in another, depending on the variability (standard error).
- Standardizing ensures that the magnitude of the test statistic reflects both the size of the difference and the reliability of the sample data.

4.1.4.2 Interpreting the Test Statistic

After calculating the test statistic, it is used to:

1. Compare with a critical value: For example, in a z-test with $\alpha = 0.05$, the critical values are -1.96 and 1.96 for a two-tailed test. If the test statistic falls beyond these values, H_0 is rejected.
2. Calculate the p-value: The p-value is derived from the distribution and reflects the probability of observing a test statistic as extreme as the one calculated if H_0 is true.

4.1.5 Critical Values and Rejection Regions

The **critical value** is a point on the distribution that separates the rejection region from the non-rejection region:

- **Rejection Region**: If the test statistic falls in this region, we reject H_0.
- **Non-Rejection Region**: If the test statistic falls here, we fail to reject H_0.

The rejection region depends on the significance level (α). For a two-tailed test with $\alpha = 0.05$, the critical values correspond to the top 2.5% and bottom 2.5% of the distribution.

4.1.6 Visualizing Hypothesis Testing

Let's create a visualization to tie these concepts together:

```r
# Parameters
alpha <- 0.05  # Significance level
df <- 29       # Degrees of freedom (for t-distribution)
t_critical <-
    qt(1 - alpha / 2, df)  # Critical value for two-tailed test

# Generate t-distribution values
t_values <- seq(-4, 4, length.out = 1000)
density <- dt(t_values, df)

# Observed test statistic
t_obs <- 2.5  # Example observed test statistic
```

```r
# Plot the t-distribution
plot(
    t_values,
    density,
    type = "l",
    lwd = 2,
    col = "blue",
    main = "Hypothesis Testing with Distribution",
    xlab = "Test Statistic (t-value)",
    ylab = "Density",
    ylim = c(0, 0.4)
)

# Shade the rejection regions
polygon(c(t_values[t_values <= -t_critical], -t_critical),
        c(density[t_values <= -t_critical], 0),
        col = "red",
        border = NA)
polygon(c(t_values[t_values >= t_critical], t_critical),
        c(density[t_values >= t_critical], 0),
        col = "red",
        border = NA)

# Add observed test statistic
points(
    t_obs,
    dt(t_obs, df),
    col = "green",
    pch = 19,
    cex = 1.5
)
text(
    t_obs,
    dt(t_obs, df) + 0.02,
    paste("Observed t:", round(t_obs, 2)),
    col = "green",
    pos = 3
)

# Highlight the critical values
abline(
    v = c(-t_critical, t_critical),
    col = "black",
    lty = 2
)
```

```r
text(
    -t_critical,
    0.05,
    paste("Critical Value:", round(-t_critical, 2)),
    pos = 4,
    col = "black"
)
text(
    t_critical,
    0.05,
    paste("Critical Value:", round(t_critical, 2)),
    pos = 4,
    col = "black"
)

# Calculate p-value
p_value <- 2 * (1 - pt(abs(t_obs), df))  # Two-tailed p-value
text(0,
     0.35,
     paste("P-value:", round(p_value, 4)),
     col = "blue",
     pos = 3)

# Annotate regions
text(-3,
     0.15,
     "Rejection Region",
     col = "red",
     pos = 3)
text(3, 0.15, "Rejection Region", col = "red", pos = 3)
text(0,
     0.05,
     "Non-Rejection Region",
     col = "blue",
     pos = 3)

# Add legend
legend(
    "topright",
    legend = c("Rejection Region", "Critical Value", "Observed Test Statistic"),
    col = c("red", "black", "green"),
    lty = c(NA, 2, NA),
    pch = c(15, NA, 19),
    bty = "n"
)
```

4.2 Key Concepts and Definitions

4.2.1 Random Sample

A random sample of size n consists of n independent observations, each drawn from the same underlying population distribution. Independence ensures that no observation influences another, and identical distribution guarantees that all observations are governed by the same probability rules.

4.2.2 Sample Statistics

4.2.2.1 Sample Mean

The sample mean is a measure of central tendency:

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$

- Example: Suppose we measure the heights of 5 individuals (in cm): $170, 165, 180, 175, 172$. The sample mean is:

$$\bar{X} = \frac{170 + 165 + 180 + 175 + 172}{5} = 172.4\,\text{cm}.$$

4.2.2.2 Sample Median

The sample median is the middle value of ordered data:

$$\tilde{x} = \begin{cases} \text{Middle observation,} & \text{if } n \text{ is odd,} \\ \text{Average of two middle observations,} & \text{if } n \text{ is even.} \end{cases}$$

4.2.2.3 Sample Variance

The sample variance measures data spread:

$$S^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}$$

4.2.2.4 Sample Standard Deviation

The sample standard deviation is the square root of the variance:

$$S = \sqrt{S^2}$$

4.2.2.5 Sample Proportions

Used for categorical data:

$$\hat{p} = \frac{X}{n} = \frac{\text{Number of successes}}{\text{Sample size}}$$

4.2.2.6 Estimators

- **Point Estimator**: A statistic ($\hat{\theta}$) used to estimate a population parameter (θ).
- **Point Estimate**:The numerical value assumed by $\hat{\theta}$ when evaluated for a given sample.
- **Unbiased Estimator**: A point estimator $\hat{\theta}$ is unbiased if $E(\hat{\theta}) = \theta$.

Examples of unbiased estimators:

- $\bar{X}$ for μ (population mean).
- S^2 for σ^2 (population variance).
- $\hat{p}$ for p (population proportion).
- $\widehat{p_1 - p_2}$ for $p_1 - p_2$ (population proportion difference)
- $\bar{X}_1 - \bar{X}_2$ for $\mu_1 - \mu_2$ (population mean difference)

Note: While S^2 is unbiased for σ^2, S is a biased estimator of σ.

4.2.3 Distribution of the Sample Mean

The sampling distribution of the mean $\bar{X}$ depends on:

1. **Population Distribution**:
   - If $X \sim N(\mu, \sigma^2)$, then $\bar{X} \sim N\left(\mu, \frac{\sigma^2}{n}\right)$.
2. **Central Limit Theorem**:
   - For large n, $\bar{X}$ approximately follows a normal distribution, regardless of the population's shape.

4.2.3.1 Standard Error of the Mean

The standard error quantifies variability in $\bar{X}$:

$$\sigma_{\bar{X}} = \frac{\sigma}{\sqrt{n}}$$

Example: - Suppose $\sigma = 10$ and $n = 25$. Then:

$$\sigma_{\bar{X}} = \frac{10}{\sqrt{25}} = 2.$$

The smaller the standard error, the more precise our estimate of the population mean.

4.3 One-Sample Inference

4.3.1 For Single Mean

Consider a scenario where

$$Y_i \sim \text{i.i.d. } N(\mu, \sigma^2),$$

where i.i.d. stands for "independent and identically distributed." This model can be expressed as:

$$Y_i = \mu + \epsilon_i,$$

where:

- $\epsilon_i \sim^{\text{i.i.d.}} N(0, \sigma^2)$,
- $E(Y_i) = \mu$,
- $\text{Var}(Y_i) = \sigma^2$,
- $\bar{y} \sim N(\mu, \sigma^2/n)$.

When σ^2 is estimated by s^2, the standardized test statistic follows a t-distribution:

$$\frac{\bar{y}-\mu}{s/\sqrt{n}} \sim t_{n-1}.$$

A $100(1-\alpha)\%$ confidence interval for μ is obtained as:

$$1-\alpha = P\left(-t_{\alpha/2;n-1} \leq \frac{\bar{y}-\mu}{s/\sqrt{n}} \leq t_{\alpha/2;n-1}\right),$$

or equivalently,

$$P\left(\bar{y} - t_{\alpha/2;n-1}\frac{s}{\sqrt{n}} \leq \mu \leq \bar{y} + t_{\alpha/2;n-1}\frac{s}{\sqrt{n}}\right).$$

The confidence interval is expressed as:

$$\bar{y} \pm t_{\alpha/2;n-1}\frac{s}{\sqrt{n}},$$

where $s/\sqrt{n}$ is the standard error of $\bar{y}$.

If the experiment were repeated many times, $100(1-\alpha)\%$ of these intervals would contain μ.

Case	Confidence Interval $100(1-\alpha)\%$	Sample Size (Confidence α, Error d)	Hypothesis Test Statistic
σ^2 known, X normal (or $n \geq 25$)	$\bar{X} \pm z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$	$n \approx \frac{z^2_{\alpha/2}\sigma^2}{d^2}$	$z = \frac{\bar{X}-\mu_0}{\sigma/\sqrt{n}}$
σ^2 unknown, X normal (or $n \geq 25$)	$\bar{X} \pm t_{\alpha/2}\frac{s}{\sqrt{n}}$	$n \approx \frac{z^2_{\alpha/2}s^2}{d^2}$	$t = \frac{\bar{X}-\mu_0}{s/\sqrt{n}}$

4.3.1.1 Power in Hypothesis Testing

Power ($\pi(\mu)$) of a hypothesis test represents the probability of correctly rejecting the null hypothesis (H_0) when it is false (i.e., when alternative hypothesis H_A is true). Formally, it is expressed as:

$$\begin{aligned}\text{Power} = \pi(\mu) &= 1 - \beta \\ &= P(\text{test rejects } H_0|\mu) \\ &= P(\text{test rejects } H_0|H_A \text{ is true}),\end{aligned}$$

where β is the probability of a Type II error (failing to reject H_0 when it is false).

To calculate this probability:

1. **Under** H_0: The distribution of the test statistic is centered around the null parameter (e.g., μ_0).
2. **Under** H_A: The test statistic is distributed differently, shifted according to the true value under H_A (e.g., μ_1).

Hence, to evaluate the power, it is crucial to determine the distribution of the test statistic under the alternative hypothesis, H_A.

Below, we derive the power for both one-sided and two-sided z-tests.

4.3.1.1.1 One-Sided z-Test Consider the hypotheses:

$$H_0 : \mu \leq \mu_0 \quad \text{vs.} \quad H_A : \mu > \mu_0$$

The power for a one-sided z-test is derived as follows:

1. The test rejects H_0 if $\bar{y} > \mu_0 + z_\alpha \frac{\sigma}{\sqrt{n}}$, where z_α is the critical value for the test at the significance level α.
2. Under the alternative hypothesis, the distribution of $\bar{y}$ is centered at μ, with standard deviation $\frac{\sigma}{\sqrt{n}}$.
3. The power is then:

$$\begin{aligned}\pi(\mu) &= P\left(\bar{y} > \mu_0 + z_\alpha \frac{\sigma}{\sqrt{n}} \middle| \mu\right) \\ &= P\left(Z > z_\alpha + \frac{\mu_0 - \mu}{\sigma/\sqrt{n}} \middle| \mu\right), \quad \text{where } Z = \frac{\bar{y} - \mu}{\sigma/\sqrt{n}} \\ &= 1 - \Phi\left(z_\alpha + \frac{(\mu_0 - \mu)\sqrt{n}}{\sigma}\right) \\ &= \Phi\left(-z_\alpha + \frac{(\mu - \mu_0)\sqrt{n}}{\sigma}\right).\end{aligned}$$

Here, we use the symmetry of the standard normal distribution: $1 - \Phi(x) = \Phi(-x)$.

Suppose we wish to show that the mean response μ under the treatment is higher than the mean response μ_0 without treatment (i.e., the treatment effect $\delta = \mu - \mu_0$ is large).

Since power is an increasing function of $\mu - \mu_0$, it suffices to find the sample size n that achieves the desired power $1 - \beta$ at $\mu = \mu_0 + \delta$. The power at $\mu = \mu_0 + \delta$ is:

$$\pi(\mu_0 + \delta) = \Phi\left(-z_\alpha + \frac{\delta\sqrt{n}}{\sigma}\right) = 1 - \beta$$

Given $\Phi(z_\beta) = 1 - \beta$, we have:

$$-z_\alpha + \frac{\delta\sqrt{n}}{\sigma} = z_\beta$$

Solving for n, we obtain:

$$n = \left(\frac{(z_\alpha + z_\beta)\sigma}{\delta}\right)^2$$

Larger sample sizes are required when:

- The sample variability is large (σ is large).
- The significance level α is small (z_α is large).
- The desired power $1 - \beta$ is large (z_β is large).
- The magnitude of the effect is small (δ is small).

In practice, δ and σ are often unknown. To estimate σ, you can:

1. Use prior studies or pilot studies.
2. Approximate σ based on the anticipated range of the observations (excluding outliers). For normally distributed data, dividing the range by 4 provides a reasonable estimate of σ.

These considerations ensure the test is adequately powered to detect meaningful effects while balancing practical constraints such as sample size.

4.3.1.1.2 Two-Sided z-Test For a two-sided test, the hypotheses are:

$$H_0 : \mu = \mu_0 \quad \text{vs.} \quad H_A : \mu \neq \mu_0$$

The test rejects H_0 if $\bar{y}$ lies outside the interval $\mu_0 \pm z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$. The power of the test is:

$$\begin{aligned}\pi(\mu) &= P\left(\bar{y} < \mu_0 - z_{\alpha/2}\frac{\sigma}{\sqrt{n}}\middle|\mu\right) + P\left(\bar{y} > \mu_0 + z_{\alpha/2}\frac{\sigma}{\sqrt{n}}\middle|\mu\right) \\ &= \Phi\left(-z_{\alpha/2} + \frac{(\mu-\mu_0)\sqrt{n}}{\sigma}\right) + \Phi\left(-z_{\alpha/2} - \frac{(\mu-\mu_0)\sqrt{n}}{\sigma}\right).\end{aligned}$$

To ensure a power of $1-\beta$ when the treatment effect $\delta = |\mu - \mu_0|$ is at least a certain value, we solve for n. Since the power function for a two-sided test is increasing and symmetric in $|\mu - \mu_0|$, it suffices to find n such that the power equals $1-\beta$ when $\mu = \mu_0 + \delta$. This gives:

$$n = \left(\frac{(z_{\alpha/2} + z_\beta)\sigma}{\delta}\right)^2$$

Alternatively, the required sample size can be determined using a confidence interval approach. For a two-sided α-level confidence interval of the form:

$$\bar{y} \pm D$$

where $D = z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$, solving for n gives:

$$n = \left(\frac{z_{\alpha/2}\sigma}{D}\right)^2$$

This value should be rounded up to the nearest integer to ensure the required precision.

```
# Generate random data and compute a 95% confidence interval
data <- rnorm(100) # Generate 100 random values
t.test(data, conf.level = 0.95) # Perform t-test with 95% confidence interval
#>
#>  One Sample t-test
#>
#> data:  data
#> t = -1.0355, df = 99, p-value = 0.303
#> alternative hypothesis: true mean is not equal to 0
#> 95 percent confidence interval:
#>  -0.29774458  0.09354984
#> sample estimates:
#>  mean of x
#> -0.1020974
```

For a one-sided hypothesis test, such as testing $H_0 : \mu \geq 30$ versus $H_a : \mu < 30$:

```r
# Perform one-sided t-test
t.test(data, mu = 30, alternative = "less")
#>
#>  One Sample t-test
#>
#> data:  data
#> t = -305.29, df = 99, p-value < 2.2e-16
#> alternative hypothesis: true mean is less than 30
#> 95 percent confidence interval:
#>        -Inf 0.06162006
#> sample estimates:
#>  mean of x
#> -0.1020974
```

When σ is unknown, you can estimate it using:

1. Prior studies or pilot studies.
2. The range of observations (excluding outliers) divided by 4, which provides a reasonable approximation for normally distributed data.

4.3.1.1.3 z-Test Summary

- For one-sided tests:

$$\pi(\mu) = \Phi\left(-z_\alpha + \frac{(\mu - \mu_0)\sqrt{n}}{\sigma}\right)$$

- For two-sided tests:

$$\pi(\mu) = \Phi\left(-z_{\alpha/2} + \frac{(\mu - \mu_0)\sqrt{n}}{\sigma}\right) + \Phi\left(-z_{\alpha/2} - \frac{(\mu - \mu_0)\sqrt{n}}{\sigma}\right)$$

Factors Affecting Power

- **Effect Size** ($\mu - \mu_0$): Larger differences between μ and μ_0 increase power.
- **Sample Size** (n): Larger n reduces the standard error, increasing power.
- **Variance** (σ^2): Smaller variance increases power.
- **Significance Level** (α): Increasing α (making the test more liberal) increases power through z_α.

4.3.1.1.4 One-Sample t-test

In hypothesis testing, calculating the power and determining the required sample size for **t-tests** are more complex than for **z-tests**. This complexity arises from the involvement of the **Student's t-distribution** and its generalized form, the **non-central t-distribution**.

The power function for a one-sample t-test can be expressed as:

$$\pi(\mu) = P\left(\frac{\bar{y} - \mu_0}{s/\sqrt{n}} > t_{n-1;\alpha} \mid \mu\right)$$

Here:

- μ_0 is the hypothesized population mean under the null hypothesis,
- $\bar{y}$ is the sample mean,
- s is the sample standard deviation,
- n is the sample size,
- $t_{n-1;\alpha}$ is the critical t-value from the Student's t-distribution with $n-1$ degrees of freedom at significance level α.

When $\mu > \mu_0$ (i.e., $\mu - \mu_0 = \delta$), the random variable

$$T = \frac{\bar{y} - \mu_0}{s/\sqrt{n}}$$

does not follow the Student's t-distribution. Instead, it follows a **non-central t-distribution** with:

- a **non-centrality parameter** $\lambda = \delta\sqrt{n}/\sigma$, where σ is the population standard deviation,
- degrees of freedom $n-1$.

Key Properties of the Power Function

- The power $\pi(\mu)$ is an increasing function of the non-centrality parameter λ.
- For $\delta = 0$ (i.e., when the null hypothesis is true), the non-central t-distribution simplifies to the regular Student's t-distribution.

To calculate the power in practice, numerical procedures (see below) or precomputed charts are typically required.

Approximate Sample Size Adjustment for t-tests

When planning a study, researchers often start with an approximation based on **z-tests** and then adjust for the specifics of the t-test. Here's the process:

1. Start with the Sample Size for a z-test

For a two-sided test:

$$n_z = \frac{\left(z_{\alpha/2} + z_\beta\right)^2 \sigma^2}{\delta^2}$$

where:

- $z_{\alpha/2}$ is the critical value from the standard normal distribution for a two-tailed test,
- z_β corresponds to the desired power $1-\beta$,
- δ is the effect size $\mu - \mu_0$,
- σ is the population standard deviation.

2. Adjust for the t-distribution

Let $v = n-1$, where n is the sample size derived from the z-test. For a two-sided t-test, the approximate sample size is:

$$n^* = \frac{\left(t_{v;\alpha/2} + t_{v;\beta}\right)^2 \sigma^2}{\delta^2}$$

Here:

- $t_{v;\alpha/2}$ and $t_{v;\beta}$ are the critical values from the Student's t-distribution for the significance level α and desired power, respectively.
- Since v depends on n^*, this process may require iterative refinement.

Notes:

1. **Approximations**: The above formulas provide an intuitive starting point but may require adjustments based on exact numerical solutions.
2. **Insights**: Power is an increasing function of:
   - the effect size δ,
   - the sample size n,
   - and a decreasing function of the population variability σ.

```r
# Example: Power calculation for a one-sample t-test
library(pwr)

# Parameters
effect_size <- 0.5  # Cohen's d
alpha <- 0.05       # Significance level
power <- 0.8        # Desired power

# Compute sample size
sample_size <-
    pwr.t.test(
        d = effect_size,
        sig.level = alpha,
        power = power,
        type = "one.sample"
    )$n
```

```r
# Print result
cat("Required sample size for one-sample t-test:",
    ceiling(sample_size),
    "\n")
#> Required sample size for one-sample t-test: 34

# Power calculation for a given sample size
calculated_power <-
    pwr.t.test(
        n = ceiling(sample_size),
        d = effect_size,
        sig.level = alpha,
        type = "one.sample"
    )$power
cat("Achieved power with computed sample size:",
    calculated_power,
    "\n")
#> Achieved power with computed sample size: 0.8077775
```

4.3.2 For Difference of Means, Independent Samples

	$100(1-\alpha)$ Confidence Interval	Hypothesis Testing Test Statistic	
When σ^2 is known	$\bar{X}_1 - \bar{X}_2 \pm z_{\alpha/2}\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}$	$z = \frac{(\bar{X}_1-\bar{X}_2)-(\mu_1-\mu_2)_0}{\sqrt{\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}}}$	
When σ^2 is unknown, Variances Assumed EQUAL	$\bar{X}_1 - \bar{X}_2 \pm t_{\alpha/2}\sqrt{s_p^2(\frac{1}{n_1} + \frac{1}{n_2})}$	$t = \frac{(\bar{X}_1-\bar{X}_2)-(\mu_1-\mu_2)_0}{\sqrt{s_p^2(\frac{1}{n_1}+\frac{1}{n_2})}}$	Pooled Variance: $s_p^2 = \frac{(n_1-1)s_1^2-(n_2-1)s_2^2}{n_1+n_2-2}$ Degrees of Freedom: $\gamma = n_1 + n_2 - 2$
When σ^2 is unknown, Variances Assumed UN-EQUAL	$\bar{X}_1 - \bar{X}_2 \pm t_{\alpha/2}\sqrt{(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2})}$	$t = \frac{(\bar{X}_1-\bar{X}_2)-(\mu_1-\mu_2)_0}{\sqrt{(\frac{s_1^2}{n_1}+\frac{s_2^2}{n_2})}}$	Degrees of Freedom: $\gamma = \frac{(\frac{s_1^2}{n_1}+\frac{s_2^2}{n_2})^2}{\frac{(\frac{s_1^2}{n_1})^2}{n_1-1}+\frac{(\frac{s_2^2}{n_2})^2}{n_2-1}}$

4.3.3 For Difference of Means, Paired Samples

Metric	Formula
Confidence Interval	$\bar{D} \pm t_{\alpha/2} \frac{s_d}{\sqrt{n}}$
Hypothesis Test Statistic	$t = \frac{\bar{D} - D_0}{s_d/\sqrt{n}}$

4.3.4 For Difference of Two Proportions

The mean of the difference between two sample proportions is given by:

$$\hat{p_1} - \hat{p_2}$$

The variance of the difference in proportions is:

$$\frac{p_1(1-p_1)}{n_1} + \frac{p_2(1-p_2)}{n_2}$$

A $100(1-\alpha)\%$ confidence interval for the difference in proportions is calculated as:

$$\hat{p_1} - \hat{p_2} \pm z_{\alpha/2} \sqrt{\frac{p_1(1-p_1)}{n_1} + \frac{p_2(1-p_2)}{n_2}}$$

where

- $z_{\alpha/2}$: The critical value from the standard normal distribution.
- $\hat{p_1}$, $\hat{p_2}$: Sample proportions.
- n_1, n_2: Sample sizes.

Sample Size for a Desired Confidence Level and Margin of Error

To achieve a margin of error d for a given confidence level, the required sample size can be estimated as follows:

1. **With Prior Estimates of** $\hat{p_1}$ and $\hat{p_2}$:

$$n \approx \frac{z_{\alpha/2}^2 \left[p_1(1-p_1) + p_2(1-p_2)\right]}{d^2}$$

2. **Without Prior Estimates** (assuming maximum variability, $\hat{p} = 0.5$):

$$n \approx \frac{z_{\alpha/2}^2}{2d^2}$$

Hypothesis Testing for Difference in Proportions

The test statistic for hypothesis testing depends on the null hypothesis:

1. **When** $(p_1 - p_2) \neq 0$:

$$z = \frac{(\hat{p_1} - \hat{p_2}) - (p_1 - p_2)_0}{\sqrt{\frac{p_1(1-p_1)}{n_1} + \frac{p_2(1-p_2)}{n_2}}}$$

2. **When** $(p_1 - p_2)_0 = 0$ (testing equality of proportions):

$$z = \frac{\hat{p_1} - \hat{p_2}}{\sqrt{\hat{p}(1-\hat{p})\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}$$

where $\hat{p}$ is the pooled sample proportion:

$$\hat{p} = \frac{x_1 + x_2}{n_1 + n_2} = \frac{n_1\hat{p_1} + n_2\hat{p_2}}{n_1 + n_2}$$

4.3.5 For Single Proportion

The $100(1-\alpha)\%$ confidence interval for a population proportion p is:

$$\hat{p} \pm z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

Sample Size Determination

- **With Prior Estimate** $(\hat{p})$:

$$n \approx \frac{z^2_{\alpha/2}\hat{p}(1-\hat{p})}{d^2}$$

- **Without Prior Estimate**:

$$n \approx \frac{z^2_{\alpha/2}}{4d^2}$$

The test statistic for $H_0 : p = p_0$ is:

$$z = \frac{\hat{p} - p_0}{\sqrt{\frac{p_0(1-p_0)}{n}}}$$

4.3.6 For Single Variance

For a sample variance s^2 with n observations, the $100(1-\alpha)\%$ confidence interval for the population variance σ^2 is:

$$\begin{aligned} 1-\alpha &= P(\chi^2_{1-\alpha/2;n-1}) \leq (n-1)s^2/\sigma^2 \leq \chi^2_{\alpha/2;n-1}) \\ &= P\left(\frac{(n-1)s^2}{\chi^2_{\alpha/2;n-1}} \leq \sigma^2 \leq \frac{(n-1)s^2}{\chi^2_{1-\alpha/2;n-1}}\right) \end{aligned}$$

Equivalently, the confidence interval can be written as:

$$\left(\frac{(n-1)s^2}{\chi^2_{\alpha/2}}, \frac{(n-1)s^2}{\chi^2_{1-\alpha/2}}\right)$$

To find confidence limits for σ, compute the square root of the interval bounds:

$$\text{Confidence Interval for } \sigma: \quad \left(\sqrt{\frac{(n-1)s^2}{\chi^2_{\alpha/2}}}, \sqrt{\frac{(n-1)s^2}{\chi^2_{1-\alpha/2}}}\right)$$

Hypothesis Testing for Variance

The test statistic for testing a null hypothesis about a population variance (σ_0^2) is:

$$\chi^2 = \frac{(n-1)s^2}{\sigma_0^2}$$

This test statistic follows a chi-squared distribution with $n-1$ degrees of freedom under the null hypothesis.

4.3.7 Non-parametric Tests

Method	**Purpose**	**Assumptions**
Sign Test	Test median	None (ordinal data sufficient)
Wilcoxon Signed Rank Test	Test symmetry around a value	Symmetry of distribution
Wald-Wolfowitz Runs Test	Test for randomness	Independent observations
Quantile (or Percentile) Test	Test specific quantile	None (ordinal data sufficient)

4.3.7.1 Sign Test

The **Sign Test** is used to test hypotheses about the median of a population, $\mu_{(0.5)}$, without assuming a specific distribution for the data. This test is ideal for small sample sizes or when normality assumptions are not met.

To test the population median, consider the hypotheses:

- Null Hypothesis: $H_0 : \mu_{(0.5)} = 0$
- Alternative Hypothesis: $H_a : \mu_{(0.5)} > 0$ (one-sided test)

Steps:

1. **Count Positive and Negative Deviations**:
   - Count observations (y_i) greater than 0: s_+ (number of positive signs).
   - Count observations less than 0: s_- (number of negative signs).
   - $s_- = n - s_+$.
2. **Decision Rule**:
   - Reject H_0 if s_+ is large (or equivalently, s_- is small).
   - To determine how large s_+ must be, use the distribution of S_+ under H_0, which is **Binomial** with $p = 0.5$.
3. **Null Distribution**:
   Under H_0, S_+ follows:
   $$S_+ \sim Binomial(n, p = 0.5)$$
4. **Critical Value**:
   Reject H_0 if:
   $$s_+ \geq b_{n,\alpha}$$
   where $b_{n,\alpha}$ is the upper α critical value of the binomial distribution.
5. **p-value Calculation**:
   Compute the p-value for the observed (one-tailed) s_+ as:
   $$\text{p-value} = P(S \geq s_+) = \sum_{i=s_+}^{n} \binom{n}{i} \left(\frac{1}{2}\right)^n$$
   Alternatively:
   $$P(S \leq s_-) = \sum_{i=0}^{s_-} \binom{n}{i} \left(\frac{1}{2}\right)^n$$

Large Sample Normal Approximation

For large n, use a normal approximation for the binomial test. Reject H_0 if:

$$s_+ \geq \frac{n}{2} + \frac{1}{2} + z_\alpha \sqrt{\frac{n}{4}}$$

where z_α is the critical value for a one-sided test.

For two-sided tests, use the maximum or minimum of s_+ and s_-:

- Test statistic: $s_{\max} = \max(s_+, s_-)$ or $s_{\min} = \min(s_+, s_-)$
- Reject H_0 if p-value is less than α, where:

$$p\text{-value} = 2 \sum_{i=s_{\max}}^{n} \binom{n}{i} \left(\frac{1}{2}\right)^n = 2 \sum_{i=0}^{s_{min}} \binom{n}{i} \left(\frac{1}{2}\right)^n$$

Equivalently, rejecting H_0 if $s_{max} \geq b_{n,\alpha/2}$.

For large n, the normal approximation uses:

$$z = \frac{s_{\max} - \frac{n}{2} - \frac{1}{2}}{\sqrt{\frac{n}{4}}}$$

Reject H_0 at α if $z \geq z_{\alpha/2}$.

Handling zeros in the data is a common issue with the Sign Test:

1. **Random Assignment**: Assign zeros randomly to either s_+ or s_- (2 researchers might get different results).
2. **Fractional Assignment**: Count each zero as 0.5 toward both s_+ and s_- (but then we could not apply the Binomial Distribution afterward).
3. **Ignore Zeros**: Ignore zeros, but note this reduces the sample size and power.

```r
# Example Data
data <- c(0.76, 0.82, 0.80, 0.79, 1.06, 0.83, -0.43, -0.34, 3.34, 2.33)

# Count positive signs
s_plus <- sum(data > 0)

# Sample size excluding zeros
n <- length(data)

# Perform a one-sided binomial test
binom.test(s_plus, n, p = 0.5, alternative = "greater")
#>
#>  Exact binomial test
#>
#> data:  s_plus and n
#> number of successes = 8, number of trials = 10, p-value = 0.05469
#> alternative hypothesis: true probability of success is greater than 0.5
#> 95 percent confidence interval:
#>  0.4930987 1.0000000
```

```
#> sample estimates:
#> probability of success
#>                    0.8
```

4.3.7.2 Wilcoxon Signed Rank Test

The **Wilcoxon Signed Rank Test** is an improvement over the Sign Test as it considers both the magnitude and direction of deviations from the null hypothesis value (e.g., 0). However, this test assumes that the data are symmetrically distributed around the median, unlike the Sign Test.

We test the following hypotheses:

$$H_0 : \mu_{(0.5)} = 0 H_a : \mu_{(0.5)} > 0$$

This example assumes no ties or duplicate observations in the data.

Procedure for the Signed Rank Test

1. **Rank the Absolute Values**:
   - Rank the observations y_i based on their absolute values.
   - Let r_i denote the rank of y_i.
   - Since there are no ties, ranks r_i are uniquely determined and form a permutation of integers $1, 2, \ldots, n$.
2. **Calculate** w_+ and w_-:
   - w_+ is the sum of the ranks corresponding to positive values of y_i.
   - w_- is the sum of the ranks corresponding to negative values of y_i.
   - By definition:
   $$w_+ + w_- = \sum_{i=1}^{n} r_i = \frac{n(n+1)}{2}$$
3. **Decision Rule**:
   - Reject H_0 if w_+ is large (or equivalently, if w_- is small).

Null Distribution of W_+

Under the null hypothesis, the distributions of W_+ and W_- are identical and symmetric. The p-value for a one-sided test is:

$$\text{p-value} = P(W \geq w_+) = P(W \leq w_-)$$

An α-level test rejects H_0 if $w_+ \geq w_{n,\alpha}$, where $w_{n,\alpha}$ is the critical value from a table of the null distribution of W_+.

For two-sided tests, use:

$$p\text{-value} = 2P(W \geq w_{max}) = 2P(W \leq w_{min})$$

Normal Approximation for Large Samples

For large n, the null distribution of W_+ can be approximated by a normal distribution:

$$z = \frac{w_+ - \frac{n(n+1)}{4} - \frac{1}{2}}{\sqrt{\frac{n(n+1)(2n+1)}{24}}}$$

The test rejects H_0 at level α if:

$$w_+ \geq \frac{n(n+1)}{4} + \frac{1}{2} + z_\alpha \sqrt{\frac{n(n+1)(2n+1)}{24}} \approx w_{n,\alpha}$$

For a two-sided test, the decision rule uses the maximum or minimum of w_+ and w_-:

- $w_{max} = \max(w_+, w_-)$
- $w_{min} = \min(w_+, w_-)$

The p-value is computed as:

$$p\text{-value} = 2P(W \geq w_{max}) = 2P(W \leq w_{min})$$

Handling Tied Ranks

If some observations $|y_i|$ have tied absolute values, assign the average rank (or "midrank") to all tied values. For example:

- Suppose $y_1 = -1$, $y_2 = 3$, $y_3 = -3$, and $y_4 = 5$.
- The ranks for $|y_i|$ are:
  - $|y_1| = 1$: $r_1 = 1$
  - $|y_2| = |y_3| = 3$: $r_2 = r_3 = \frac{2+3}{2} = 2.5$
  - $|y_4| = 5$: $r_4 = 4$

```r
# Example Data
data <- c(0.76, 0.82, 0.80, 0.79, 1.06, 0.83, -0.43, -0.34, 3.34, 2.33)

# Perform Wilcoxon Signed Rank Test (exact test)
wilcox_exact <- wilcox.test(data, exact = TRUE)

# Display results
```

```
wilcox_exact
#>
#>  Wilcoxon signed rank exact test
#>
#> data:  data
#> V = 52, p-value = 0.009766
#> alternative hypothesis: true location is not equal to 0
```

For large samples, you can use the normal approximation by setting `exact = FALSE`:

```
# Perform Wilcoxon Signed Rank Test (normal approximation)
wilcox_normal <- wilcox.test(data, exact = FALSE)

# Display results
wilcox_normal
#>
#>  Wilcoxon signed rank test with continuity correction
#>
#> data:  data
#> V = 52, p-value = 0.01443
#> alternative hypothesis: true location is not equal to 0
```

4.3.7.3 Wald-Wolfowitz Runs Test

The **Runs Test** is a non-parametric test used to examine the randomness of a sequence. Specifically, it tests whether the order of observations in a sequence is random. This test is useful in detecting non-random patterns, such as trends, clustering, or periodicity.

The hypotheses for the Runs Test are:

- Null Hypothesis: H_0: The sequence is random.
- Alternative Hypothesis: H_a: The sequence is not random.

A **run** is a sequence of consecutive observations of the same type. For example: - In the binary sequence `+ + - - + - + +`, there are **5 runs**: `++`, `--`, `+`, `-`, `++`.

Runs can be formed based on any classification criteria, such as:

- Positive vs. Negative values
- Above vs. Below the median
- Success vs. Failure in binary outcomes

Test Statistic

1. **Number of Runs** (R):
   The observed number of runs in the sequence.

2. **Expected Number of Runs** ($E[R]$):
   Under the null hypothesis of randomness, the expected number of runs is:

$$E[R] = \frac{2n_1 n_2}{n_1 + n_2} + 1$$

   where:

   - n_1: Number of observations in the first category (e.g., positives).
   - n_2: Number of observations in the second category (e.g., negatives).
   - $n = n_1 + n_2$: Total number of observations.

3. **Variance of Runs** ($\text{Var}[R]$):
   The variance of the number of runs is given by:

$$\text{Var}[R] = \frac{2n_1 n_2 (2n_1 n_2 - n)}{n^2(n-1)}$$

4. **Standardized Test Statistic** (z):
   For large samples ($n \geq 20$), the test statistic is approximately normally distributed:

$$z = \frac{R - E[R]}{\sqrt{\text{Var}[R]}}$$

Decision Rule

- Compute the z-value and compare it to the critical value of the standard normal distribution.
- For a significance level α:
  - Reject H_0 if $|z| \geq z_{\alpha/2}$ (two-sided test).
  - Reject H_0 if $z \geq z_\alpha$ or $z \leq -z_\alpha$ for one-sided tests.

Steps for Conducting a Runs Test:

1. Classify the data into two groups (e.g., above/below median, positive/negative).
2. Count the total number of runs (R).
3. Compute $E[R]$ and $\text{Var}[R]$ based on n_1 and n_2.
4. Compute the z-value for the observed number of runs.
5. Compare the z-value to the critical value to decide whether to reject H_0.

For a numerical dataset where the test is based on values above and below the median:

```r
# Example dataset
data <- c(1.2, -0.5, 3.4, -1.1, 2.8, -0.8, 4.5, 0.7)

library(randtests)
# Perform Runs Test (above/below median)
runs.test(data)
```

```
#>
#>  Runs Test
#>
#> data:  data
#> statistic = 2.2913, runs = 8, n1 = 4, n2 = 4, n = 8, p-value = 0.02195
#> alternative hypothesis: nonrandomness
```

The output of the `runs.test` function includes:

- **Observed Runs**: The actual number of runs in the sequence.
- **Expected Runs**: The expected number of runs under H_0.
- **p-value**: The probability of observing a number of runs as extreme as the observed one under H_0.
- If the p-value is less than α, reject H_0 and conclude that the sequence is not random.

Limitations of the Runs Test

- The test assumes that observations are independent.
- For small sample sizes, the test may have limited power.
- Ties in the data must be resolved by a predefined rule (e.g., treating ties as belonging to one group or excluding them).

4.3.7.4 Quantile (or Percentile) Test

The **Quantile Test** (also called the Percentile Test) is a non-parametric test used to evaluate whether the proportion of observations falling within a specific quantile matches the expected proportion under the null hypothesis. This test is useful for assessing the distribution of data when specific quantiles (e.g., medians or percentiles) are of interest.

Suppose we want to test whether the true proportion of data below a specified quantile q matches a given probability p. The hypotheses are:

- Null Hypothesis: H_0: The true proportion is equal to p.
- Alternative Hypothesis: H_a: The true proportion is not equal to p (two-sided), greater than p (right-tailed), or less than p (left-tailed).

Test Statistic

The test statistic is based on the observed count of data points below the specified quantile.

1. **Observed Count** (k):
   The number of data points y_i such that $y_i \leq q$.

2. **Expected Count** ($E[k]$):
   The expected number of observations below the quantile q under H_0 is:

$$E[k] = n \cdot p$$

3. **Variance**:
   Under the binomial distribution, the variance is:

$$\text{Var}[k] = n \cdot p \cdot (1 - p)$$

4. **Standardized Test Statistic** (z):
   For large n, the test statistic is approximately normally distributed:

$$z = \frac{k - E[k]}{\sqrt{\text{Var}[k]}} = \frac{k - n \cdot p}{\sqrt{n \cdot p \cdot (1 - p)}}$$

Decision Rule

1. Compute the z-value for the observed count.
2. Compare the z-value to the critical value of the standard normal distribution:
   - For a two-sided test, reject H_0 if $|z| \geq z_{\alpha/2}$.
   - For a one-sided test, reject H_0 if $z \geq z_\alpha$ (right-tailed) or $z \leq -z_\alpha$ (left-tailed).

Alternatively, calculate the p-value and reject H_0 if the p-value $\leq \alpha$.

Suppose we have a dataset and want to test whether the proportion of observations below the 50th percentile (median) matches the expected value of $p = 0.5$.

```r
# Example data
data <- c(12, 15, 14, 10, 13, 11, 14, 16, 15, 13)

# Define the quantile to test
quantile_value <- quantile(data, 0.5) # Median
p <- 0.5                              # Proportion under H0

# Count observed values below or equal to the quantile
k <- sum(data <= quantile_value)

# Sample size
n <- length(data)

# Expected count under H0
expected_count <- n * p

# Variance
```

```r
variance <- n * p * (1 - p)

# Test statistic (z-value)
z <- (k - expected_count) / sqrt(variance)

# Calculate p-value for two-sided test
p_value <- 2 * (1 - pnorm(abs(z)))

# Output results
list(
  quantile_value = quantile_value,
  observed_count = k,
  expected_count = expected_count,
  z_value = z,
  p_value = p_value
)
#> $quantile_value
#>  50%
#> 13.5
#>
#> $observed_count
#> [1] 5
#>
#> $expected_count
#> [1] 5
#>
#> $z_value
#> [1] 0
#>
#> $p_value
#> [1] 1
```

For a one-sided test (e.g., testing whether the proportion is greater than p):

```r
# Calculate one-sided p-value
p_value_one_sided <- 1 - pnorm(z)

# Output one-sided p-value
p_value_one_sided
#> [1] 0.5
```

Interpretation of Results

- **p-value**: If the p-value is less than α, reject H_0 and conclude that the proportion of observations below the quantile deviates significantly from p.

- **Quantile Test Statistic (z)**: The z-value indicates how many standard deviations the observed count is from the expected count under the null hypothesis. Large positive or negative z values suggest non-random deviations.

Assumptions of the Test

1. Observations are independent.
2. The sample size is large enough for the normal approximation to the binomial distribution to be valid ($n \cdot p \geq 5$ and $n \cdot (1-p) \geq 5$).

Limitations of the Test

- For small sample sizes, the normal approximation may not hold. In such cases, exact binomial tests are more appropriate.
- The test assumes that the quantile used (e.g., the median) is well-defined and correctly calculated from the data.

4.4 Two-Sample Inference

4.4.1 For Means

Suppose we have two sets of observations:

- $y_1, \dots, y_{n_y}$
- $x_1, \dots, x_{n_x}$

These are random samples from two independent populations with means μ_y and μ_x and variances σ_y^2 and σ_x^2. Our goal is to compare μ_y and μ_x or test whether $\sigma_y^2 = \sigma_x^2$.

4.4.1.1 Large Sample Tests

If n_y and n_x are large (≥ 30), the Central Limit Theorem allows us to make the following assumptions:

- **Expectation**:
$$E(\bar{y} - \bar{x}) = \mu_y - \mu_x$$
- **Variance**:
$$\mathrm{Var}(\bar{y} - \bar{x}) = \frac{\sigma_y^2}{n_y} + \frac{\sigma_x^2}{n_x}$$

The test statistic is:

$$Z = \frac{\bar{y} - \bar{x} - (\mu_y - \mu_x)}{\sqrt{\frac{\sigma_y^2}{n_y} + \frac{\sigma_x^2}{n_x}}} \sim N(0,1)$$

For large samples, replace variances with their unbiased estimators s_y^2 and s_x^2, yielding the same large sample distribution.

Confidence Interval

An approximate $100(1-\alpha)\%$ confidence interval for $\mu_y - \mu_x$ is:

$$\bar{y} - \bar{x} \pm z_{\alpha/2}\sqrt{\frac{s_y^2}{n_y} + \frac{s_x^2}{n_x}}$$

Hypothesis Test

Testing:

$$H_0 : \mu_y - \mu_x = \delta_0 \quad \text{vs.} \quad H_a : \mu_y - \mu_x \neq \delta_0$$

The test statistic:

$$z = \frac{\bar{y} - \bar{x} - \delta_0}{\sqrt{\frac{s_y^2}{n_y} + \frac{s_x^2}{n_x}}}$$

Reject H_0 at the α-level if:

$$|z| > z_{\alpha/2}$$

If $\delta_0 = 0$, this tests whether the two means are equal.

```r
# Large sample test
y <- c(10, 12, 14, 16, 18)
x <- c(9, 11, 13, 15, 17)

# Mean and variance
mean_y <- mean(y)
mean_x <- mean(x)
var_y <- var(y)
var_x <- var(x)
n_y <- length(y)
n_x <- length(x)

# Test statistic
z <- (mean_y - mean_x) / sqrt(var_y / n_y + var_x / n_x)
p_value <- 2 * (1 - pnorm(abs(z)))

list(z = z, p_value = p_value)
#> $z
```

```
#> [1] 0.5
#>
#> $p_value
#> [1] 0.6170751
```

4.4.1.2 Small Sample Tests

If the samples are small, assume the data come from independent normal distributions:

- $y_i \sim N(\mu_y, \sigma_y^2)$
- $x_i \sim N(\mu_x, \sigma_x^2)$

We can do inference based on the Student's T Distribution, where we have 2 cases:

- Equal Variances
- Unequal Variances

Assumption	Tests	Plots
Independence and Identically Distributed (i.i.d.) Observations	Test for serial correlation	
Independence Between Samples	Correlation Coefficient	Scatterplot
Normality	See Normality Assessment	See Normality Assessment
Equality of Variances	1. F-Test 2. Levene's Test 3. Modified Levene Test (Brown-Forsythe Test) 4. Bartlett's Test	1. Boxplots with overlayed means 2. Residuals spread plots

4.4.1.2.1 Equal Variances Assumptions

1. Independence and Identically Distributed (i.i.d.) Observations

Assume that observations in each sample are i.i.d., which implies:

$$var(\bar{y}) = \frac{\sigma_y^2}{n_y}, \quad var(\bar{x}) = \frac{\sigma_x^2}{n_x}$$

2. Independence Between Samples

The samples are assumed to be independent, meaning no observation from one sample influences observations from the other. This independence allows us to write:

$$\begin{aligned} var(\bar{y} - \bar{x}) &= var(\bar{y}) + var(\bar{x}) - 2cov(\bar{y}, \bar{x}) \\ &= var(\bar{y}) + var(\bar{x}) \\ &= \frac{\sigma_y^2}{n_y} + \frac{\sigma_x^2}{n_x} \end{aligned}$$

This calculation assumes $cov(\bar{y}, \bar{x}) = 0$ due to the independence between the samples.

3. Normality Assumption

We assume that the underlying populations are normally distributed. This assumption justifies the use of the Student's T Distribution, which is critical for hypothesis testing and constructing confidence intervals.

4. Equality of Variances

If the population variances are equal, i.e., $\sigma_y^2 = \sigma_x^2 = \sigma^2$, then s_y^2 and s_x^2 are both unbiased estimators of σ^2. This allows us to pool the variances.

The pooled variance estimator is calculated as:

$$s^2 = \frac{(n_y - 1)s_y^2 + (n_x - 1)s_x^2}{(n_y - 1) + (n_x - 1)}$$

The pooled variance estimate has degrees of freedom equal to:

$$df = (n_y + n_x - 2)$$

Test Statistic

The test statistic is:

$$T = \frac{\bar{y} - \bar{x} - (\mu_y - \mu_x)}{s\sqrt{\frac{1}{n_y} + \frac{1}{n_x}}} \sim t_{n_y + n_x - 2}$$

Confidence Interval

A $100(1-\alpha)\%$ confidence interval for $\mu_y - \mu_x$ is:

$$\bar{y} - \bar{x} \pm t_{n_y+n_x-2,\alpha/2} \cdot s\sqrt{\frac{1}{n_y} + \frac{1}{n_x}}$$

Hypothesis Test

Testing:

$$H_0 : \mu_y - \mu_x = \delta_0 \quad \text{vs.} \quad H_a : \mu_y - \mu_x \neq \delta_0$$

Reject H_0 if:

$$|T| > t_{n_y+n_x-2,\alpha/2}$$

```
# Small sample test with equal variance
t_test_equal <- t.test(y, x, var.equal = TRUE)
t_test_equal
#>
#>  Two Sample t-test
#>
#> data:  y and x
#> t = 0.5, df = 8, p-value = 0.6305
#> alternative hypothesis: true difference in means is not equal to 0
#> 95 percent confidence interval:
#>  -3.612008  5.612008
#> sample estimates:
#> mean of x mean of y
#>        14        13
```

4.4.1.2.2 Unequal Variances Assumptions

1. Independence and Identically Distributed (i.i.d.) Observations

Assume that observations in each sample are i.i.d., which implies:

$$var(\bar{y}) = \frac{\sigma_y^2}{n_y}, \quad var(\bar{x}) = \frac{\sigma_x^2}{n_x}$$

2. Independence Between Samples

The samples are assumed to be independent, meaning no observation from one sample influences observations from the other. This independence allows us to write:

$$\begin{aligned} var(\bar{y} - \bar{x}) &= var(\bar{y}) + var(\bar{x}) - 2cov(\bar{y}, \bar{x}) \\ &= var(\bar{y}) + var(\bar{x}) \\ &= \frac{\sigma_y^2}{n_y} + \frac{\sigma_x^2}{n_x} \end{aligned}$$

This calculation assumes $cov(\bar{y}, \bar{x}) = 0$ due to the independence between the samples.

3. Normality Assumption

We assume that the underlying populations are normally distributed. This assumption justifies the use of the Student's T Distribution, which is critical for hypothesis testing and constructing confidence intervals.

4. Unequal Variances

$\sigma_y^2 \neq \sigma_x^2$

Test Statistic

The test statistic is:

$$T = \frac{\bar{y} - \bar{x} - (\mu_y - \mu_x)}{\sqrt{\frac{s_y^2}{n_y} + \frac{s_x^2}{n_x}}}$$

Degrees of Freedom (Welch-Satterthwaite Approximation) (Satterthwaite, 1946)

The degrees of freedom are approximated by:

$$v = \frac{\left(\frac{s_y^2}{n_y} + \frac{s_x^2}{n_x}\right)^2}{\frac{\left(\frac{s_y^2}{n_y}\right)^2}{n_y - 1} + \frac{\left(\frac{s_x^2}{n_x}\right)^2}{n_x - 1}}$$

Since v is fractional, truncate to the nearest integer.

Confidence Interval

A $100(1 - \alpha)\%$ confidence interval for $\mu_y - \mu_x$ is:

$$\bar{y} - \bar{x} \pm t_{v,\alpha/2} \sqrt{\frac{s_y^2}{n_y} + \frac{s_x^2}{n_x}}$$

Hypothesis Test

Testing:

$$H_0 : \mu_y - \mu_x = \delta_0 \quad \text{vs.} \quad H_a : \mu_y - \mu_x \neq \delta_0$$

Reject H_0 if:

$$|T| > t_{v,\alpha/2}$$

where

$$t = \frac{\bar{y} - \bar{x} - \delta_0}{\sqrt{s_y^2/n_y + s_x^2/n_x}}$$

```
# Small sample test with unequal variance
t_test_unequal <- t.test(y, x, var.equal = FALSE)
t_test_unequal
#>
#>  Welch Two Sample t-test
#>
#> data:  y and x
#> t = 0.5, df = 8, p-value = 0.6305
#> alternative hypothesis: true difference in means is not equal to 0
#> 95 percent confidence interval:
#>  -3.612008  5.612008
#> sample estimates:
#> mean of x mean of y
#>        14        13
```

4.4.2 For Variances

To compare the variances of two independent samples, we can use the **F-test**. The test statistic is defined as:

$$F_{ndf,ddf} = \frac{s_1^2}{s_2^2}$$

where $s_1^2 > s_2^2$, $ndf = n_1 - 1$, and $ddf = n_2 - 1$ are the numerator and denominator degrees of freedom, respectively.

4.4.2.1 F-Test

The hypotheses for the F-test are:

$$H_0 : \sigma_y^2 = \sigma_x^2 \quad \text{(equal variances)} H_a : \sigma_y^2 \neq \sigma_x^2 \quad \text{(unequal variances)}$$

The test statistic is:

$$F = \frac{s_y^2}{s_x^2}$$

where s_y^2 and s_x^2 are the sample variances of the two groups.

Decision Rule

Reject H_0 if:

- $F > F_{n_y-1,n_x-1,\alpha/2}$ (upper critical value), or
- $F < F_{n_y-1,n_x-1,1-\alpha/2}$ (lower critical value).

Here:

- $F_{n_y-1,n_x-1,\alpha/2}$ and $F_{n_y-1,n_x-1,1-\alpha/2}$ are the critical points of the **F-distribution**, with $n_y - 1$ and $n_x - 1$ degrees of freedom.

Assumptions

- The F-test requires that the data in both groups follow a **normal distribution**.
- The F-test is sensitive to deviations from normality (e.g., heavy-tailed distributions). If the normality assumption is violated, it may lead to an inflated Type I error rate (false positives).

Limitations and Alternatives

1. **Sensitivity to Non-Normality**:
   - When data have long-tailed distributions (positive kurtosis), the F-test may produce misleading results.
   - To assess normality, see Normality Assessment.
2. **Nonparametric Alternatives**:
   - If the normality assumption is not met, use robust tests such as the Modified Levene Test (Brown-Forsythe Test), which compares group variances based on medians instead of means.

```r
# Load iris dataset
data(iris)

# Subset data for two species
irisVe <- iris$Petal.Width[iris$Species == "versicolor"]
irisVi <- iris$Petal.Width[iris$Species == "virginica"]

# Perform F-test
f_test <- var.test(irisVe, irisVi)

# Display results
```

```
f_test
#>
#>  F test to compare two variances
#>
#> data:  irisVe and irisVi
#> F = 0.51842, num df = 49, denom df = 49, p-value = 0.02335
#> alternative hypothesis: true ratio of variances is not equal to 1
#> 95 percent confidence interval:
#>  0.2941935 0.9135614
#> sample estimates:
#> ratio of variances
#>          0.5184243
```

4.4.2.2 Levene's Test

Levene's Test is a robust method for testing the equality of variances across multiple groups. Unlike the F-test, it is less sensitive to departures from normality and is particularly useful for handling non-normal distributions and datasets with outliers. The test works by analyzing the deviations of individual observations from their group mean or median.

Test Procedure

1. Compute the absolute deviations of each observation from its group mean or median:
   - For group y:
   $$d_{y,i} = |y_i - \text{Central Value}_y|$$
   - For group x:
   $$d_{x,j} = |x_j - \text{Central Value}_x|$$
   - The "central value" can be either the **mean** (classic Levene's test) or the **median** (Modified Levene Test (Brown-Forsythe Test) variation, more robust for non-normal data).
2. Perform a one-way ANOVA on the absolute deviations to test for differences in group variances.

Hypotheses

- Null Hypothesis (H_0): All groups have equal variances.
- Alternative Hypothesis (H_a): At least one group has a variance different from the others.

Test Statistic

The Levene test statistic is calculated as an ANOVA on the absolute deviations. Let:

- k: Number of groups,

- n_i: Number of observations in group i,
- n: Total number of observations.

The test statistic is:

$$W = \frac{(n-k)\sum_{i=1}^{k} n_i(\bar{d}_i - \bar{d})^2}{(k-1)\sum_{i=1}^{k}\sum_{j=1}^{n_i}(d_{i,j} - \bar{d}_i)^2}$$

where:

- $d_{i,j}$: Absolute deviations within group i,
- $\bar{d}_i$: Mean of the absolute deviations for group i,
- $\bar{d}$: Overall mean of the absolute deviations.

Under the null hypothesis, $W \sim F_{k-1,n-k}$.

Decision Rule

- Compute the test statistic W.
- Reject H_0 at significance level α if:

$$W > F_{k-1,n-k,\alpha}$$

```
# Load required package
library(car)

# Perform Levene's Test (absolute deviations from the mean)
levene_test_mean <- leveneTest(Petal.Width ~ Species, data = iris)

# Perform Levene's Test (absolute deviations from the median)
levene_test_median <-
    leveneTest(Petal.Width ~ Species, data = iris, center = median)

# Display results
levene_test_mean
#> Levene's Test for Homogeneity of Variance (center = median)
#>        Df F value    Pr(>F)
#> group   2  19.892 2.261e-08 ***
#>       147
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
levene_test_median
#> Levene's Test for Homogeneity of Variance (center = median)
#>        Df F value    Pr(>F)
#> group   2  19.892 2.261e-08 ***
#>       147
```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The output includes:

- **Df**: Degrees of freedom for the numerator and denominator.
- **F-value**: The computed value of the test statistic W.
- **p-value**: The probability of observing such a value under the null hypothesis.
- If the p-value is less than α, reject H_0 and conclude that the group variances are significantly different.
- Otherwise, fail to reject H_0 and conclude there is no evidence of a difference in variances.

Advantages of Levene's Test

1. **Robustness**:
   - Handles non-normal data and outliers better than the F-test.
2. **Flexibility**:
   - By choosing the center value (mean or median), it can adapt to different data characteristics:
     - Use the mean for symmetric distributions.
     - Use the median for non-normal or skewed data.
3. **Versatility**:
   - Applicable to comparing variances across more than two groups, unlike the Modified Levene Test (Brown-Forsythe Test), which is limited to two groups.

4.4.2.3 Modified Levene Test (Brown-Forsythe Test)

The **Modified Levene Test** is a robust alternative to the F-test for comparing variances between two groups. Instead of using squared deviations (as in the F-test), this test considers the **absolute deviations** from the median, making it less sensitive to non-normal data and long-tailed distributions. It is, however, still appropriate for normally distributed data.

For each sample, compute the absolute deviations from the median:

$$d_{y,i} = |y_i - y_{.5}| \quad \text{and} \quad d_{x,i} = |x_i - x_{.5}|$$

Let:

- $\bar{d}_y$ and $\bar{d}_x$ be the means of the absolute deviations for groups y and x, respectively.

The test statistic is:

$$t_L^* = \frac{\bar{d}_y - \bar{d}_x}{s\sqrt{\frac{1}{n_y} + \frac{1}{n_x}}}$$

where the pooled variance s^2 is:

$$s^2 = \frac{\sum_{i=1}^{n_y}(d_{y,i} - \bar{d}_y)^2 + \sum_{j=1}^{n_x}(d_{x,j} - \bar{d}_x)^2}{n_y + n_x - 2}$$

Assumptions

1. **Constant Variance of Error Terms**:
   The test assumes equal error variances in each group under the null hypothesis.
2. **Moderate Sample Size**:
   The approximation $t_L^* \sim t_{n_y+n_x-2}$ holds well for moderate or large sample sizes.

Decision Rule

- Compute t_L^* using the formula above.
- Reject the null hypothesis of equal variances if:

$$|t_L^*| > t_{n_y+n_x-2;\alpha/2}$$

This is equivalent to applying a two-sample t-test to the absolute deviations.

```r
# Absolute deviations from the median
dVe <- abs(irisVe - median(irisVe))
dVi <- abs(irisVi - median(irisVi))

# Perform t-test on absolute deviations
levene_test <- t.test(dVe, dVi, var.equal = TRUE)

# Display results
levene_test
#>
#>  Two Sample t-test
#>
#> data:  dVe and dVi
#> t = -2.5584, df = 98, p-value = 0.01205
#> alternative hypothesis: true difference in means is not equal to 0
```

```
#> 95 percent confidence interval:
#>  -0.12784786 -0.01615214
#> sample estimates:
#> mean of x mean of y
#>     0.154     0.226
```

For small sample sizes, use the unequal variance t-test directly on the original data as a robust alternative:

```
# Small sample t-test with unequal variances
small_sample_test <- t.test(irisVe, irisVi, var.equal = FALSE)

# Display results
small_sample_test
#>
#>  Welch Two Sample t-test
#>
#> data:  irisVe and irisVi
#> t = -14.625, df = 89.043, p-value < 2.2e-16
#> alternative hypothesis: true difference in means is not equal to 0
#> 95 percent confidence interval:
#>  -0.7951002 -0.6048998
#> sample estimates:
#> mean of x mean of y
#>     1.326     2.026
```

4.4.2.4 Bartlett's Test

The **Bartlett's Test** is a statistical procedure for testing the equality of variances across multiple groups. It assumes that the data in each group are normally distributed and is sensitive to deviations from normality. When the assumption of normality holds, Bartlett's Test is more powerful than Levene's Test.

Hypotheses for Bartlett's Test

- Null Hypothesis (H_0): All groups have equal variances.
- Alternative Hypothesis (H_a): At least one group has a variance different from the others.

The test statistic for Bartlett's Test is:

$$B = \frac{(n-k)\log(S_p^2) - \sum_{i=1}^{k}(n_i - 1)\log(S_i^2)}{1 + \frac{1}{3(k-1)}\left(\sum_{i=1}^{k}\frac{1}{n_i-1} - \frac{1}{n-k}\right)}$$

Where:

- k: Number of groups,
- n_i: Number of observations in group i,
- $n = \sum_{i=1}^{k} n_i$: Total number of observations,
- S_i^2: Sample variance of group i,
- S_p^2: Pooled variance, given by:

$$S_p^2 = \frac{\sum_{i=1}^{k}(n_i - 1)S_i^2}{n - k}$$

Under the null hypothesis, the test statistic $B \sim \chi^2_{k-1}$.

Assumptions

1. **Normality**: The data in each group must follow a normal distribution.
2. **Independence**: Observations within and between groups must be independent.
3. **Equal Sample Sizes (Optional)**: Bartlett's Test is more robust if sample sizes are approximately equal.

Decision Rule

- Compute the test statistic B.
- Compare B to the critical value of the Chi-Square distribution at α and $k - 1$ degrees of freedom.
- Reject H_0 if:

$$B > \chi^2_{k-1,\alpha}$$

Alternatively, use the p-value:

- Reject H_0 if the p-value $\leq \alpha$.

```
# Perform Bartlett's Test
bartlett_test <- bartlett.test(Petal.Width ~ Species, data = iris)

# Display results
bartlett_test
#>
#>  Bartlett test of homogeneity of variances
#>
#> data:  Petal.Width by Species
#> Bartlett's K-squared = 39.213, df = 2, p-value = 3.055e-09
```

The output includes:

- **Bartlett's K-squared**: The value of the test statistic B.
- **df**: Degrees of freedom $(k - 1)$, where k is the number of groups.

- **p-value**: The probability of observing such a value of B under H_0.
- If the p-value is less than α, reject H_0 and conclude that the variances are significantly different across groups.
- If the p-value is greater than α, fail to reject H_0 and conclude that there is no significant evidence of variance differences.

Limitations of Bartlett's Test

1. **Sensitivity to Non-Normality**:
   Bartlett's Test is highly sensitive to departures from normality. Even slight deviations can lead to misleading results.
2. **Not Robust to Outliers**:
   Outliers can disproportionately affect the test result.
3. **Alternatives**:
   If the normality assumption is violated, use robust alternatives like:
   - Levene's Test (absolute deviations)
   - Modified Levene Test (Brown-Forsythe Test) (median-based absolute deviations)

Advantages of Bartlett's Test

- **High Power**: Bartlett's Test is more powerful than robust alternatives when the normality assumption holds.
- **Simple Implementation**: The test is easy to perform and interpret.

4.4.3 Power

To evaluate the power of a test, we consider the situation where the variances are equal across groups:

$$\sigma_y^2 = \sigma_x^2 = \sigma^2$$

Under the assumption of equal variances, we take **equal sample sizes** from both groups, i.e., $n_y = n_x = n$.

Hypotheses for One-Sided Testing

We are testing:

$$H_0 : \mu_y - \mu_x \leq 0 \quad \text{vs.} \quad H_a : \mu_y - \mu_x > 0$$

Test Statistic

The α-level **z-test** rejects H_0 if the test statistic:

$$z = \frac{\bar{y} - \bar{x}}{\sigma\sqrt{\frac{2}{n}}} > z_\alpha$$

where:

- $\bar{y}$ and $\bar{x}$ are the sample means,
- σ is the common standard deviation,
- z_α is the critical value from the standard normal distribution.

Power Function

The **power** of the test, denoted as $\pi(\mu_y - \mu_x)$, is the probability of correctly rejecting H_0 when $\mu_y - \mu_x$ is some specified value. Under the alternative hypothesis, the power function is:

$$\pi(\mu_y - \mu_x) = \Phi\left(-z_\alpha + \frac{\mu_y - \mu_x}{\sigma}\sqrt{\frac{n}{2}}\right)$$

where:

- Φ is the cumulative distribution function (CDF) of the standard normal distribution,
- $\frac{\mu_y - \mu_x}{\sigma}\sqrt{\frac{n}{2}}$ represents the standardized effect size.

Determining the Required Sample Size

To achieve a desired power of $1 - \beta$ when the true difference is δ (the smallest difference of interest), we solve for the required sample size n. The power equation is:

$$\Phi\left(-z_\alpha + \frac{\delta}{\sigma}\sqrt{\frac{n}{2}}\right) = 1 - \beta$$

Rearranging for n, the required sample size is:

$$n = \frac{2\sigma^2}{\delta^2}\left(z_\alpha + z_\beta\right)^2$$

where:

- σ: The common standard deviation,
- z_α: The critical value for the Type I error rate α (one-sided test),
- z_β: The critical value for the Type II error rate β (related to power $1-\beta$),
- δ: The minimum detectable difference between the means.

```r
# Parameters
alpha <- 0.05   # Significance level
beta <- 0.2     # Type II error rate (1 - Power = 0.2)
sigma <- 1      # Common standard deviation
delta <- 0.5    # Minimum detectable difference

# Critical values
z_alpha <- qnorm(1 - alpha)
z_beta <- qnorm(1 - beta)

# Sample size calculation
n <- (2 * sigma ^ 2 * (z_alpha + z_beta) ^ 2) / delta ^ 2

# Output the required sample size (per group)
ceiling(n)
#> [1] 50
```

Sample Size for Two-Sided Tests

For a **two-sided test**, replace z_α with $z_{\alpha/2}$ to account for the two-tailed critical region:

$$n = 2\left(\frac{\sigma(z_{\alpha/2} + z_\beta)}{\delta}\right)^2$$

This ensures that the test has the required power $1-\beta$ to detect a difference of size δ between the means at significance level α.

Adjustment for the Exact t-Test

When conducting an exact **two-sample t-test** for small sample sizes, the sample size calculation involves the **non-central t-distribution**. An approximate correction can be applied using the critical values from the t-distribution instead of the z-distribution.

The adjusted sample size is:

$$n^* = 2\left(\frac{\sigma(t_{2n-2;\alpha/2} + t_{2n-2;\beta})}{\delta}\right)^2$$

Where:

- $t_{2n-2;\alpha/2}$: The critical value for the t-distribution with $2n-2$ degrees of freedom for significance level $\alpha/2$,
- $t_{2n-2;\beta}$: The critical value for the t-distribution with $2n-2$ degrees of freedom for power $1-\beta$.

This correction adjusts for the increased variability of the t-distribution, especially important for small sample sizes.

```
# Parameters
alpha <- 0.05    # Significance level
power <- 0.8     # Desired power
sigma <- 1       # Common standard deviation
delta <- 0.5     # Minimum detectable difference

# Calculate sample size for two-sided test
sample_size <-
    power.t.test(
        delta = delta,
        sd = sigma,
        sig.level = alpha,
        power = power,
        type = "two.sample",
        alternative = "two.sided"
    )

# Display results
sample_size
#>
#>      Two-sample t test power calculation
#>
#>               n = 63.76576
#>           delta = 0.5
#>              sd = 1
#>       sig.level = 0.05
#>           power = 0.8
#>     alternative = two.sided
#>
#> NOTE: n is number in *each* group
```

Key Insights

1. **Z-Test vs. T-Test**:
   For large samples, the normal approximation (z-test) works well. For small samples, the t-test correction using the t-distribution is essential.

2. **Effect of Power and Significance Level**:

   - Increasing power $(1-\beta)$ or decreasing α requires larger sample sizes.
   - A smaller minimum detectable difference (δ) also requires a larger sample size.

3. **Two-Sided Tests**:
   Two-sided tests require larger sample sizes compared to one-sided tests

due to the split critical region.

Formula Summary

Test Type	Formula for Sample Size
One-Sided Test	$n = 2\left(\frac{\sigma(z_\alpha + z_\beta)}{\delta}\right)^2$
Two-Sided Test	$n = 2\left(\frac{\sigma(z_{\alpha/2} + z_\beta)}{\delta}\right)^2$
Approximate t-Test	$n^* = 2\left(\frac{\sigma(t_{2n-2;\alpha/2} + t_{2n-2;\beta})}{\delta}\right)^2$

4.4.4 Matched Pair Designs

In **matched pair designs**, two treatments are compared by measuring responses for the same subjects under both treatments. This ensures that the effects of subject-to-subject variability are minimized, as each subject serves as their own control.

We have two treatments, and the data are structured as follows:

Subject	Treatment A	Treatment B	Difference
1	y_1	x_1	$d_1 = y_1 - x_1$
2	y_2	x_2	$d_2 = y_2 - x_2$
...	...	...	...
n	y_n	x_n	$d_n = y_n - x_n$

Here:

- y_i represents the observation under Treatment A,
- x_i represents the observation under Treatment B,
- $d_i = y_i - x_i$ is the difference for subject i.

Assumptions

1. Observations y_i and x_i are measured for the same subjects, inducing correlation.
2. The differences d_i are independent and identically distributed (iid), and follow a normal distribution:

$$d_i \sim N(\mu_D, \sigma_D^2)$$

Mean and Variance of the Difference

The mean difference μ_D and the variance σ_D^2 are given by:

$$\mu_D = E(y_i - x_i) = \mu_y - \mu_x$$

$$\sigma_D^2 = \text{Var}(y_i - x_i) = \text{Var}(y_i) + \text{Var}(x_i) - 2 \cdot \text{Cov}(y_i, x_i)$$

- If the **covariance** between y_i and x_i is **positive** (a typical case), the variance of the differences σ_D^2 is **reduced** compared to the independent sample case.
- This is the key benefit of **Matched Pair Designs**: reduced variability increases the precision of estimates.

Sample Statistics

For the differences $d_i = y_i - x_i$:

- The sample mean of the differences:

$$\bar{d} = \frac{1}{n} \sum_{i=1}^{n} d_i = \bar{y} - \bar{x}$$

- The sample variance of the differences:

$$s_d^2 = \frac{1}{n-1} \sum_{i=1}^{n} (d_i - \bar{d})^2$$

Once the data are converted into differences d_i, the problem reduces to **one-sample inference**. We can use tests and confidence intervals (CIs) for the mean of a single sample.

Hypothesis Test

We test the following hypotheses:

$$H_0 : \mu_D = 0 \quad \text{vs.} \quad H_a : \mu_D \neq 0$$

The test statistic is:

$$t = \frac{\bar{d}}{s_d/\sqrt{n}} \sim t_{n-1}$$

where n is the number of subjects.

- Reject H_0 at significance level α if:

$$|t| > t_{n-1,\alpha/2}$$

Confidence Interval

A $100(1-\alpha)\%$ confidence interval for μ_D is:

$$\bar{d} \pm t_{n-1,\alpha/2} \cdot \frac{s_d}{\sqrt{n}}$$

```
# Sample data
treatment_a <- c(85, 90, 78, 92, 88)
treatment_b <- c(80, 86, 75, 89, 85)

# Compute differences
differences <- treatment_a - treatment_b

# Perform one-sample t-test on the differences
t_test <- t.test(differences, mu = 0, alternative = "two.sided")

# Display results
t_test
#>
#>  One Sample t-test
#>
#> data:  differences
#> t = 9, df = 4, p-value = 0.0008438
#> alternative hypothesis: true mean is not equal to 0
#> 95 percent confidence interval:
#>  2.489422 4.710578
#> sample estimates:
#> mean of x
#>       3.6
```

The output includes:

- **t-statistic**: The calculated test statistic for the matched pairs.
- **p-value**: The probability of observing such a difference under the null hypothesis.
- **Confidence Interval**: The range of plausible values for the mean difference μ_D.
- If the p-value is less than α, reject H_0 and conclude that there is a significant difference between the two treatments.
- If the confidence interval does not include 0, this supports the conclusion of a significant difference.

Key Insights

1. **Reduced Variability**: Positive correlation between paired observations reduces the variance of the differences, increasing test power.
2. **Use of Differences**: The paired design converts the data into a single-sample problem for inference.
3. **Robustness**: The paired t-test assumes normality of the differences d_i. For larger n, the Central Limit Theorem ensures robustness to non-normality.

Matched pair designs are a powerful way to control for subject-specific variability, leading to more precise comparisons between treatments.

4.4.5 Nonparametric Tests for Two Samples

For Matched Pair Designs or independent samples where normality cannot be assumed, we use **nonparametric tests**. These tests do not assume any specific distribution of the data and are robust alternatives to parametric methods.

Stochastic Order and Location Shift

Suppose Y and X are random variables with cumulative distribution functions (CDFs) F_Y and F_X. Then Y is **stochastically larger** than X if, for all real numbers u:

$$P(Y > u) \geq P(X > u) \quad \text{(equivalently, } F_Y(u) \leq F_X(u)\text{)}.$$

If the two distributions differ only in their **location parameters**, say θ_y and θ_x, then we can frame the relationship as:

$$Y > X \quad \text{if} \quad \theta_y > \theta_x.$$

We test the following hypotheses:

- **Two-Sided Hypothesis**:

$$H_0 : F_Y = F_X \quad \text{vs.} \quad H_a : F_Y \neq F_X$$

- **Upper One-Sided Hypothesis**:

$$H_0 : F_Y = F_X \quad \text{vs.} \quad H_a : F_Y < F_X$$

- **Lower One-Sided Hypothesis**:

$$H_0 : F_Y = F_X \quad \text{vs.} \quad H_a : F_Y > F_X$$

We generally avoid the completely non-directional alternative $H_a : F_Y \neq F_X$ because it allows arbitrary differences between the distributions, without requiring one distribution to be stochastically larger than the other.

Nonparametric Tests

When the focus is on whether the two distributions differ **only in location parameters**, two equivalent nonparametric tests are commonly used:

1. Wilcoxon Signed Rank Test
2. Mann-Whitney U Test

Both tests are mathematically equivalent and test whether one sample is systematically larger than the other.

4.4.5.1 Wilcoxon Rank-Sum Test

The **Wilcoxon Rank Test** is a nonparametric test used to compare two independent samples to assess whether their distributions differ in location. It is based on the ranks of the combined observations rather than their actual values.

Procedure

1. **Combine and Rank Observations**:
   Combine all $n = n_y + n_x$ observations (from both groups) into a single dataset and rank them in ascending order. If ties exist, assign the average rank to tied values.

2. **Calculate Rank Sums**:
   Compute the sum of ranks for each group:

   - w_y: Sum of the ranks for group y (sample 1),
   - w_x: Sum of the ranks for group x (sample 2).
     By definition:
     $$w_y + w_x = \frac{n(n+1)}{2}$$

3. **Test Statistic**:
   The test focuses on the rank sum w_y. Reject H_0 if w_y is **large** (indicating y systematically has larger values) or equivalently, if w_x is **small**.

4. **Null Distribution**:
   Under H_0 (no difference between groups), all possible arrangements of ranks among y and x are equally likely. The total number of possible rank arrangements is:
   $$\frac{(n_y + n_x)!}{n_y!\, n_x!}$$

5. **Computational Considerations**:

   - For small samples, the exact null distribution of the rank sums can be calculated.

- For large samples, an **approximate normal distribution** can be used.

Hypotheses

- Null Hypothesis (H_0): The two samples come from identical distributions.
- Alternative Hypothesis (H_a): The two samples come from different distributions, or one distribution is systematically larger.
- **Two-Sided Test**:
$$H_a : F_Y \neq F_X$$
- **One-Sided Test**:
$$H_a : F_Y > F_X \quad \text{or} \quad H_a : F_Y < F_X$$

```r
# Subset data for two species
irisVe <- iris$Petal.Width[iris$Species == "versicolor"]
irisVi <- iris$Petal.Width[iris$Species == "virginica"]

# Perform Wilcoxon Rank Test (approximate version, large sample)
wilcox_result <- wilcox.test(
    irisVe,
    irisVi,
    alternative = "two.sided", # Two-sided test
    conf.level = 0.95,         # Confidence level
    exact = FALSE,             # Approximate test for large samples
    correct = TRUE             # Apply continuity correction
)

# Display results
wilcox_result
#>
#>  Wilcoxon rank sum test with continuity correction
#>
#> data:  irisVe and irisVi
#> W = 49, p-value < 2.2e-16
#> alternative hypothesis: true location shift is not equal to 0
```

The output of `wilcox.test` includes:

- **W**: The test statistic, which is the smaller of the two rank sums.
- **p-value**: The probability of observing such a difference in rank sums under H_0.
- **Alternative Hypothesis**: Specifies whether the test was one-sided or two-sided.

- **Confidence Interval** (if applicable): Provides a range for the difference in medians.

Decision Rule

1. Reject H_0 at significance level α if the p-value $\leq \alpha$.
2. For large samples, compare the test statistic to a critical value from the normal approximation.

Key Features

1. **Robustness**:
   The test does not require assumptions of normality and is robust to outliers.
2. **Distribution-Free**:
   It evaluates whether two samples differ in location without assuming a specific distribution.
3. **Rank-Based**:
   It uses the ranks of the observations, which makes it scale-invariant (resistant to data transformation).

Computational Considerations

- For small sample sizes, the **exact distribution** of the rank sums is used.
- For large sample sizes, the **normal approximation** with continuity correction is applied for computational efficiency.

4.4.5.2 Mann-Whitney U Test

The **Mann-Whitney U Test** is a nonparametric test used to compare two independent samples. It evaluates whether one sample tends to produce larger observations than the other, based on pairwise comparisons. The test does not assume normality and is robust to outliers.

Procedure

1. **Pairwise Comparisons**:
   Compare each observation y_i from sample Y with each observation x_j from sample X.

   - Let u_y be the number of pairs where $y_i > x_j$.
   - Let u_x be the number of pairs where $y_i < x_j$.

   By definition:

   $$u_y + u_x = n_y n_x$$

   where n_y is the sample size for group Y, and n_x is the sample size for group X.

2. **Test Statistic**:
   Reject H_0 if u_y is **large** (or equivalently, if u_x is **small**).

   The **Mann-Whitney U Test** and **Wilcoxon Rank-Sum Test** are related through the rank sums:

$$u_y = w_y - \frac{n_y(n_y+1)}{2}, \quad u_x = w_x - \frac{n_x(n_x+1)}{2}$$

   Here, w_y and w_x are the rank sums for groups Y and X, respectively.

Hypotheses

- Null Hypothesis (H_0): The two samples come from identical distributions.
- Alternative Hypothesis (H_a):
  - **Upper One-Sided**: $F_Y < F_X$ (Sample Y is stochastically larger).
  - **Lower One-Sided**: $F_Y > F_X$ (Sample X is stochastically larger).
  - **Two-Sided**: $F_Y \neq F_X$ (Distributions differ in location).

Test Statistic for Large Samples

For large sample sizes n_y and n_x, the null distribution of U can be approximated by a normal distribution with:

- **Mean**:
$$E(U) = \frac{n_y n_x}{2}$$
- **Variance**:
$$\mathrm{Var}(U) = \frac{n_y n_x (n_y + n_x + 1)}{12}$$

The standardized test statistic z is:

$$z = \frac{u_y - \frac{n_y n_x}{2} - \frac{1}{2}}{\sqrt{\frac{n_y n_x (n_y + n_x + 1)}{12}}}$$

The test rejects H_0 at level α if:

$$z \geq z_\alpha \quad \text{(one-sided)} \quad \text{or} \quad |z| \geq z_{\alpha/2} \quad \text{(two-sided)}.$$

For the **two-sided test**, we use:

- $u_{\max} = \max(u_y, u_x)$, and
- $u_{\min} = \min(u_y, u_x)$.

The p-value is given by:

$$p\text{-value} = 2P(U \geq u_{\max}) = 2P(U \leq u_{\min}).$$

When $y_i = x_j$ (ties), assign a value of 1/2 to both u_y and u_x for that pair. While the exact sampling distribution differs slightly when ties exist, the **large sample normal approximation** remains reasonable.

```
# Subset data for two species
irisVe <- iris$Petal.Width[iris$Species == "versicolor"]
irisVi <- iris$Petal.Width[iris$Species == "virginica"]

# Perform Mann-Whitney U Test
mann_whitney <- wilcox.test(
    irisVe, irisVi,
    alternative = "two.sided",
    conf.level = 0.95,
    exact = FALSE,   # Approximate test for large samples
    correct = TRUE   # Apply continuity correction
)

# Display results
mann_whitney
#>
#>  Wilcoxon rank sum test with continuity correction
#>
#> data:  irisVe and irisVi
#> W = 49, p-value < 2.2e-16
#> alternative hypothesis: true location shift is not equal to 0
```

Decision Rule

1. Reject H_0 if the p-value is less than α.
2. For large samples, check whether $z \ge z_{\alpha}$ (one-sided) or $|z| \ge z_{\alpha/2}$ (two-sided).

Key Insights

1. **Robustness**: The Mann-Whitney U Test does not assume normality and is robust to outliers.
2. **Relationship to Wilcoxon Test**: The test is equivalent to the **Wilcoxon Rank-Sum Test** but formulated differently (based on pairwise comparisons).
3. **Large Sample Approximation**: For large n_y and n_x, the test statistic U follows an approximate normal distribution, simplifying computation.

4. **Handling Ties**: Ties are accounted for by assigning fractional contributions to u_y and u_x.

4.5 Categorical Data Analysis

Categorical Data Analysis is used when the outcome variables are **categorical**.

- **Nominal Variables**: Categories have no logical order (e.g., sex: male, female).
- **Ordinal Variables**: Categories have a logical order, but the relative distances between values are not well defined (e.g., small, medium, large).

In categorical data, we often analyze how the distribution of one variable changes with the levels of another variable. For example, row percentages may differ across columns in a contingency table.

4.5.1 Association Tests

4.5.1.1 Small Samples

4.5.1.1.1 Fisher's Exact Test For small samples, the approximate tests based on the asymptotic normality of $\hat{p}_1 - \hat{p}_2$ (the difference in proportions) do not hold. In such cases, we use **Fisher's Exact Test** to evaluate:

- Null Hypothesis (H_0): $p_1 = p_2$ (no association between variables),
- Alternative Hypothesis (H_a): $p_1 \neq p_2$ (an association exists).

Assumptions

1. X_1 and X_2 are independent **Binomial random variables**:
   - $X_1 \sim \text{Binomial}(n_1, p_1)$,
   - $X_2 \sim \text{Binomial}(n_2, p_2)$.
2. x_1 and x_2 are the observed values (successes in each sample).
3. Total sample size is $n = n_1 + n_2$.
4. Total successes are $m = x_1 + x_2$.

By conditioning on m, the total number of successes, the number of successes in sample 1 follows a **Hypergeometric distribution**.

Test Statistic

To test $H_0 : p_1 = p_2$ against $H_a : p_1 \neq p_2$, we use the test statistic:

$$Z^2 = \left(\frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\hat{p}(1-\hat{p})\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}} \right)^2 \sim \chi^2_{1,\alpha}$$

where:

- $\hat{p}_1$ and $\hat{p}_2$ are the observed proportions of successes in samples 1 and 2,
- $\hat{p}$ is the pooled proportion:

$$\hat{p} = \frac{x_1 + x_2}{n_1 + n_2},$$

- $\chi^2_{1,\alpha}$ is the upper α critical value of the **Chi-squared distribution** with 1 degree of freedom.

Fisher's Exact Test can be extended to a **contingency table** setting to test whether the observed frequencies differ significantly from the expected frequencies under the null hypothesis of no association.

```
# Create a 2x2 contingency table
data_table <- matrix(c(8, 2, 1, 5), nrow = 2, byrow = TRUE)
colnames(data_table) <- c("Success", "Failure")
rownames(data_table) <- c("Group 1", "Group 2")

# Display the table
data_table
#>         Success Failure
#> Group 1       8       2
#> Group 2       1       5

# Perform Fisher's Exact Test
fisher_result <- fisher.test(data_table)

# Display the results
fisher_result
#>
#>  Fisher's Exact Test for Count Data
#>
#> data:  data_table
#> p-value = 0.03497
#> alternative hypothesis: true odds ratio is not equal to 1
#> 95 percent confidence interval:
#>     1.008849 1049.791446
#> sample estimates:
#> odds ratio
#>   15.46969
```

The output of `fisher.test()` includes:

- **p-value**: The probability of observing such a contingency table under the null hypothesis.
- **Alternative Hypothesis**: Indicates whether the test is two-sided or one-sided.
- If the p-value is less than α, reject H_0 and conclude that there is a significant association between the two variables.

4.5.1.1.2 Exact Chi-Square Test For small samples where the normal approximation does not apply, we can compute the **exact Chi-Square test** by using Fisher's Exact Test or Monte Carlo simulation methods.

The Chi-Square test statistic in the 2x2 table is:

$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$

where:

- O_{ij}: Observed frequency in cell (i, j),
- E_{ij}: Expected frequency under the null hypothesis,
- r: Number of rows,
- c: Number of columns.

4.5.1.2 Large Samples

4.5.1.2.1 Pearson Chi-Square Test The **Pearson Chi-Square Test** is commonly used to test whether there is an association between two categorical variables. It compares the observed counts in a contingency table to the expected counts under the null hypothesis.

The test statistic is:

$$\chi^2 = \sum_{\text{all cells}} \frac{(\text{observed} - \text{expected})^2}{\text{expected}}$$

The test is applied in settings where multiple proportions or frequencies are compared across independent surveys or experiments.

- **Null Hypothesis (H_0)**: The observed data are consistent with the expected values (no association or no deviation from a model).
- **Alternative Hypothesis (H_a)**: The observed data differ significantly from the expected values.

Characteristics of the Test

1. **Validation of Models**:
   In some cases, H_0 represents the model whose validity is being tested. The goal is not necessarily to reject the model but to check whether the data are consistent with it. Deviations may be due to random chance.

2. **Strength of Association**:
   The Chi-Square Test detects whether an association exists but does **not** measure the strength of the association. For measuring strength, metrics like Cramér's V or the Phi coefficient should be used.

3. **Effect of Sample Size**:
   - The Chi-Square statistic reflects sample size. If the sample size is doubled (e.g., duplicating observations), the χ^2 statistic will also double, even though the strength of the association remains unchanged.
   - This sensitivity can sometimes lead to detecting significant results that are not practically meaningful.

4. **Expected Cell Frequencies**:
   - The test is not appropriate if more than **20% of the cells** in a contingency table have expected frequencies less than 5.
   - For small sample sizes, Fisher's Exact Test or exact p-values should be used instead.

---

1. **Test for a Single Proportion**
   We test whether the observed proportion of successes equals 0.5.

$$H_0 : p_J = 0.5 H_a : p_J < 0.5$$

```r
# Observed data
july.x <- 480
july.n <- 1000
# Test for single proportion
prop.test(
  x = july.x,
  n = july.n,
  p = 0.5,
  alternative = "less",
  correct = FALSE
)
#>
#>  1-sample proportions test without continuity correction
#>
#> data:  july.x out of july.n, null probability 0.5
#> X-squared = 1.6, df = 1, p-value = 0.103
#> alternative hypothesis: true p is less than 0.5
```

```
#> 95 percent confidence interval:
#>  0.0000000 0.5060055
#> sample estimates:
#>    p
#> 0.48
```

2. **Test for Equality of Proportions Between Two Groups**: We test whether the proportions of successes in July and September are equal.

$$H_0 : p_J = p_S H_a : p_j \neq p_S$$

```
# Observed data for two groups
sept.x <- 704
sept.n <- 1600
# Test for equality of proportions
prop.test(
  x = c(july.x, sept.x),
  n = c(july.n, sept.n),
  correct = FALSE
)
#>
#>  2-sample test for equality of proportions without continuity correction
#>
#> data:  c(july.x, sept.x) out of c(july.n, sept.n)
#> X-squared = 3.9701, df = 1, p-value = 0.04632
#> alternative hypothesis: two.sided
#> 95 percent confidence interval:
#>  0.0006247187 0.0793752813
#> sample estimates:
#> prop 1 prop 2
#>   0.48   0.44
```

Comparison of Proportions for Multiple Groups

	Experiment 1	Experiment 2	...	Experiment k
Number of successes	x_1	x_2	...	x_k
Number of failures	$n_1 - x_1$	$n_2 - x_2$	...	$n_k - x_k$
Total	n_1	n_2	...	n_k

We test the null hypothesis:

$$H_0 : p_1 = p_2 = \cdots = p_k$$

against the alternative that at least one proportion differs.

Pooled Proportion

Assuming H_0 is true, we estimate the common value of the probability of success as:

$$\hat{p} = \frac{x_1 + x_2 + \cdots + x_k}{n_1 + n_2 + \cdots + n_k}.$$

The **expected counts** under H_0 are:

Success	$n_1\hat{p}$	$n_2\hat{p}$	...	$n_k\hat{p}$
Failure	$n_1(1-\hat{p})$	$n_2(1-\hat{p})$	...	$n_k(1-\hat{p})$
	n_1	n_2		n_k

The test statistic is:

$$\chi^2 = \sum_{\text{all cells}} \frac{(\text{observed} - \text{expected})^2}{\text{expected}}$$

with $k-1$ degrees of freedom.

Two-Way Contingency Tables

When categorical data are cross-classified, we create a two-way table of observed counts.

	1	2	...	j	...	c	Row Total
1	n_{11}	n_{12}	...	n_{1j}	...	n_{1c}	$n_{1.}$
2	n_{21}	n_{22}	...	n_{2j}	...	n_{2c}	$n_{2.}$
...	...	...	...	...	...	...	...
r	n_{r1}	n_{r2}	...	n_{rj}	...	n_{rc}	$n_{r.}$
Column Total	$n_{.1}$	$n_{.2}$	...	$n_{.j}$	...	$n_{.c}$	$n_{..}$

Sampling Designs

1. **Design 1: Total Sample Size Fixed**
   - A single random sample of size n is drawn from the population.
   - Units are cross-classified into r rows and c columns. Both row and column totals are random variables.
   - The cell counts n_{ij} follow a **multinomial distribution** with probabilities p_{ij} such that:
   $$\sum_{i=1}^{r}\sum_{j=1}^{c} p_{ij} = 1.$$
   - Let $p_{ij} = P(X = i, Y = j)$ be the joint probability, where X is the row variable and Y is the column variable.
   - **Null Hypothesis of Independence**:
   $$H_0 : p_{ij} = p_{i.}p_{.j}, \quad \text{where } p_{i.} = P(X = i) \text{ and } p_{.j} = P(Y = j).$$
   - **Alternative Hypothesis**:
   $$H_a : p_{ij} \neq p_{i.}p_{.j}.$$

2. **Design 2: Row Totals Fixed**
   - Random samples of sizes $n_1, n_2, \dots, n_r$ are drawn independently from r row populations.
   - The row totals $n_{i.}$ are fixed, but column totals are random.
   - Counts in each row follow independent **multinomial distributions**.
   - The null hypothesis assumes that the conditional probabilities of the column variable Y are the same across all rows:
   $$H_0 : p_{ij} = P(Y = j|X = i) = p_j \quad \text{for all } i \text{ and } j.$$
   - Alternatively:
   $$H_0 : (p_{i1}, p_{i2}, \dots, p_{ic}) = (p_1, p_2, \dots, p_c) \quad \text{for all } i.$$
   - **Alternative Hypothesis**:
   $$H_a : (p_{i1}, p_{i2}, \dots, p_{ic}) \text{ are not the same for all } i.$$

Design	Total Sample Size Fixed	Row Totals Fixed
Scenario	A single dataset or experiment where all observations are collected together as one sample.	Observations are collected separately for each row, with fixed totals for each row population.
Example	Survey with 100 respondents randomly selected, recording responses based on two categorical variables (e.g., age group and gender).	Stratified survey with specific numbers of individuals sampled from predefined groups (e.g., 30 males, 40 females, 30 non-binary).
Why This Design?	- Models situations where the total number of observations is fixed. - Both row and column categories emerge randomly. - Tests for **independence** between two categorical variables (row and column).	- Models scenarios where sampling occurs independently within predefined strata or groups. - Tests for **homogeneity** of column proportions across rows, ignoring differences in total counts between rows.
Practical Use Case	**Market Research**: Do customer demographics (rows) and purchase behavior (columns) show a dependence? - **Biology**: Is there an association between species (rows) and habitat types (columns)?	- **Public Health**: Are smoking rates (columns) consistent across age groups (rows)? - **Education**: Do pass rates (columns) differ across schools (rows), controlling for the number of students in each school?

Why Both Designs?

1. **Real-World Sampling Constraints**:
   - Sometimes, you have control over row totals (e.g., fixed group sizes in stratified sampling).
   - Other times, you collect data without predefined group sizes, and totals emerge randomly.
2. **Different Null Hypotheses**:
   - Design 1 tests whether two variables are **independent** (e.g., does one variable predict the other?).
   - Design 2 tests whether column proportions are **homogeneous** across groups (e.g., are the groups similar?).

```r
# Sampling Design 1: Total Sample Size Fixed
# Parameters for the multinomial distribution
r <- 3  # Number of rows
```

```r
c <- 4  # Number of columns
n <- 100  # Total sample size
p <- matrix(c(0.1, 0.2, 0.1, 0.1,
              0.05, 0.15, 0.05, 0.1,
              0.05, 0.05, 0.025, 0.075), nrow = r, byrow = TRUE)

# Generate a single random sample
set.seed(123)  # For reproducibility
n_ij <- rmultinom(1, size = n, prob = as.vector(p))

# Reshape into a contingency table
contingency_table_fixed_total <- matrix(n_ij, nrow = r, ncol = c, byrow = TRUE)
rownames(contingency_table_fixed_total) <- paste0("Row", 1:r)
colnames(contingency_table_fixed_total) <- paste0("Col", 1:c)

# Hypothesis testing (Chi-squared test of independence)
chisq_test_fixed_total <- chisq.test(contingency_table_fixed_total)

# Display results
print("Contingency Table (Total Sample Size Fixed):")
#> [1] "Contingency Table (Total Sample Size Fixed):"
print(contingency_table_fixed_total)
#>      Col1 Col2 Col3 Col4
#> Row1    8    6    4   24
#> Row2   18    1    9    7
#> Row3    2    7    5    9
print("Chi-squared Test Results:")
#> [1] "Chi-squared Test Results:"
print(chisq_test_fixed_total)
#>
#>  Pearson's Chi-squared test
#>
#> data:  contingency_table_fixed_total
#> X-squared = 28.271, df = 6, p-value = 8.355e-05
```

- All counts in the contingency table come from a single multinomial sample where both row and column totals are random.
- **Conclusion**: Reject Null. The data suggests significant dependence between row and column variables.

```r
# Sampling Design 2: Row Totals Fixed
# Parameters for the fixed row totals
n_row <- c(30, 40, 30)  # Row totals
c <- 4  # Number of columns
p_col <- c(0.25, 0.25, 0.25, 0.25)  # Common column probabilities under H0
```

```r
# Generate independent multinomial samples for each row
set.seed(123)  # For reproducibility
row_samples <- lapply(n_row, function(size) t(rmultinom(1, size, prob = p_col)))

# Combine into a contingency table
contingency_table_fixed_rows <- do.call(rbind, row_samples)
rownames(contingency_table_fixed_rows) <- paste0("Row", 1:length(n_row))
colnames(contingency_table_fixed_rows) <- paste0("Col", 1:c)

# Hypothesis testing (Chi-squared test of homogeneity)
chisq_test_fixed_rows <- chisq.test(contingency_table_fixed_rows)

# Display results
print("Contingency Table (Row Totals Fixed):")
#> [1] "Contingency Table (Row Totals Fixed):"
print(contingency_table_fixed_rows)
#>      Col1 Col2 Col3 Col4
#> Row1    6   10    7    7
#> Row2   13   13    4   10
#> Row3    8   10    6    6
print("Chi-squared Test Results:")
#> [1] "Chi-squared Test Results:"
print(chisq_test_fixed_rows)
#>
#>  Pearson's Chi-squared test
#>
#> data:  contingency_table_fixed_rows
#> X-squared = 3.2069, df = 6, p-value = 0.7825
```

- Row totals are fixed, and column counts within each row follow independent multinomial distributions.
- **Conclusion**: Fail to reject the null. The data does not provide evidence to suggest differences in column probabilities across rows.

Why Are the Results Different?

1. **Data Generation Differences**:
   - In **Design 1**, the entire table is treated as a single multinomial sample. This introduces dependencies between counts in the table.
   - In **Design 2**, rows are generated independently, and only the column probabilities are tested for consistency across rows.
2. **Null Hypotheses**:
   - **Design 1** tests independence between row and column variables

(more restrictive).

- **Design 2** tests homogeneity of column probabilities across rows (less restrictive).

Interpretation

- The results are **not directly comparable** because the null hypotheses are different:
  - **Design 1** focuses on whether rows and columns are independent across the entire table.
  - **Design 2** focuses on whether column distributions are consistent across rows.
- **Real-World Implication**:
  - If you are testing for independence (e.g., whether two variables are unrelated), use Design 1.
  - If you are testing for consistency across groups (e.g., whether proportions are the same across categories), use Design 2.

Takeaways

- The tests use the same statistical machinery (Chi-squared test), but their interpretations differ based on the experimental design and null hypothesis.
- For the same dataset, differences in assumptions can lead to different conclusions.

4.5.1.2.2 Chi-Square Test for Independence The expected frequencies $\hat{e}_{ij}$ under the null hypothesis are:

$$\hat{e}_{ij} = \frac{n_{i.} n_{.j}}{n_{..}},$$

where $n_{i.}$ and $n_{.j}$ are the row and column totals, respectively, and $n_{..}$ is the total sample size.

The test statistic is:

$$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(n_{ij} - \hat{e}_{ij})^2}{\hat{e}_{ij}} \sim \chi^2_{(r-1)(c-1)}.$$

We reject H_0 at significance level α if:

$$\chi^2 > \chi^2_{(r-1)(c-1),\alpha}.$$

Notes on the Pearson Chi-Square Test

- **Purpose**: Test for association or independence between two categorical variables.
- **Sensitivity to Sample Size**: The χ^2 statistic is proportional to sample size. Doubling the sample size doubles χ^2 even if the strength of the association remains unchanged.
- **Assumption on Expected Frequencies**: The test is not valid when more than 20% of the expected cell counts are less than 5. In such cases, exact tests are preferred.

```
# Create a contingency table
data_table <- matrix(c(30, 10, 20, 40), nrow = 2, byrow = TRUE)
colnames(data_table) <- c("Category 1", "Category 2")
rownames(data_table) <- c("Group 1", "Group 2")

# Display the table
print(data_table)
#>         Category 1 Category 2
#> Group 1         30         10
#> Group 2         20         40

# Perform Chi-Square Test
chi_result <- chisq.test(data_table)

# Display results
chi_result
#>
#>  Pearson's Chi-squared test with Yates' continuity correction
#>
#> data:  data_table
#> X-squared = 15.042, df = 1, p-value = 0.0001052
```

The output includes:

- Chi-Square Statistic (χ^2): The test statistic measuring the deviation between observed and expected counts.
- **p-value**: The probability of observing such a deviation under H_0.
- **Degrees of Freedom**: $(r-1)(c-1)$ for an $r \times c$ table.
- **Expected Frequencies**: The table of expected counts under H_0.
- If the p-value is less than α, reject H_0 and conclude that there is a significant association between the row and column variables.

4.5.1.3 Key Takeaways

Test	Purpose	Key Features	Sample Size Suitability	Statistical Assumptions
Fisher's Exact Test	Tests association between two categorical variables in a **2x2 table**.	- Computes exact p-values. - Does not rely on asymptotic assumptions. - Handles small sample sizes.	**Small sample sizes**	- Observations are independent. - Fixed marginal totals. - No normality assumption.
Exact Chi-Square Test	Tests association in larger contingency tables using exact methods.	- Generalization of Fisher's Exact Test. - Avoids asymptotic assumptions. - Suitable for small to medium datasets.	**Small to medium sample sizes**	- Observations are independent. - Marginal totals may not be fixed. - No normality assumption.
Pearson Chi-Square Test	Tests discrepancies between observed and expected frequencies.	- Most common chi-square-based test. - Includes independence and goodness-of-fit tests. - Relies on asymptotic assumptions.	**Large sample sizes**	- Observations are independent. - Expected cell frequencies 5. - Test statistic follows a chi-square distribution asymptotically.

Test	Purpose	Key Features	Sample Size Suitability	Statistical Assumptions
Chi-Square Test for Independence	Tests independence between two categorical variables in a contingency table.	- Application of Pearson Chi-Square Test. - Same assumptions as asymptotic chi-square tests. - Often used for larger contingency tables.	**Medium to large sample sizes**	- Observations are independent. - Expected cell frequencies 5. - Random sampling.

1. **Fisher's Exact Test** is specialized for small samples and fixed margins (2x2 tables).
2. **Exact Chi-Square Test** is a broader version of Fisher's for larger tables but avoids asymptotic approximations.
3. **Pearson Chi-Square Test** is the general framework, and its applications include:
   - Goodness-of-fit testing.
   - Testing independence (same as the Chi-Square Test for Independence).
4. **Chi-Square Test for Independence** is a specific application of the Pearson Chi-Square Test.

In essence:

- **Fisher's Exact Test** and **Exact Chi-Square Test** are precise methods for small datasets.
- **Pearson Chi-Square Test** and **Chi-Square Test for Independence** are interchangeable terms in many contexts, focusing on larger datasets.

4.5.2 Ordinal Association

Ordinal association refers to a relationship between two variables where the levels of one variable exhibit a consistent pattern of increase or decrease in re-

sponse to the levels of the other variable. This type of association is particularly relevant when dealing with ordinal variables, which have naturally ordered categories, such as ratings ("poor", "fair", "good", "excellent") or income brackets ("low", "medium", "high").

For example:

- As customer satisfaction ratings increase from "poor" to "excellent," the likelihood of recommending a product may also increase (positive ordinal association).
- Alternatively, as stress levels move from "low" to "high," job performance may tend to decrease (negative ordinal association).

Key Characteristics of Ordinal Association

1. **Logical Ordering of Levels**: The levels of both variables must follow a logical sequence. For instance, "small," "medium," and "large" are logically ordered, whereas categories like "blue," "round," and "tall" lack inherent order and are unsuitable for ordinal association.
2. **Monotonic Trends**: The association is typically monotonic, meaning that as one variable moves in a specific direction, the other variable tends to move in a consistent direction (either increasing or decreasing).
3. **Tests for Ordinal Association**: Specialized statistical tests assess ordinal association, focusing on how the rankings of one variable relate to those of the other. These tests require the data to respect the ordinal structure of both variables.

Practical Considerations

When using these tests, keep in mind:

- **Ordinal Data Handling**: Ensure that the data respects the ordinal structure (e.g., categories are correctly ranked and coded).
- **Sample Size**: Larger sample sizes provide more reliable estimates and stronger test power.
- **Contextual Relevance**: Interpret results within the context of the data and the research question. For example, a significant Spearman's correlation does not imply causation but rather a consistent trend.

4.5.2.1 Mantel-Haenszel Chi-square Test

The Mantel-Haenszel Chi-square Test is a statistical tool for evaluating ordinal associations, particularly when the data consists of multiple 2×2 contingency tables that examine the same association under varying conditions or strata. Unlike measures of association such as correlation coefficients, this test does not quantify the strength of the association but rather evaluates whether an association exists after controlling for stratification.

The Mantel-Haenszel Test is applicable to $2 \times 2 \times K$ contingency tables, where K represents the number of strata. Each stratum is a 2×2 table corresponding to different conditions or subgroups.

For each stratum k, let the marginal totals of the table be:

- $n_{.1k}$: Total observations in column 1
- $n_{.2k}$: Total observations in column 2
- $n_{1.k}$: Total observations in row 1
- $n_{2.k}$: Total observations in row 2
- $n_{..k}$: Total observations in the entire table

The observed cell count in row 1 and column 1 is denoted n_{11k}. Given the marginal totals, the sampling distribution of n_{11k} follows a hypergeometric distribution.

Under the assumption of conditional independence:

The expected value of n_{11k} is:

$$m_{11k} = E(n_{11k}) = \frac{n_{1.k} n_{.1k}}{n_{..k}}$$

The variance of n_{11k} is:

$$var(n_{11k}) = \frac{n_{1.k} n_{2.k} n_{.1k} n_{.2k}}{n_{..k}^2 (n_{..k} - 1)}$$

Mantel and Haenszel proposed the test statistic:

$$M^2 = \frac{\left(|\sum_k n_{11k} - \sum_k m_{11k}| - 0.5\right)^2}{\sum_k var(n_{11k})} \sim \chi_1^2$$

where

- The 0.5 adjustment, known as a continuity correction, improves the approximation to the χ^2 distribution.
- The test statistic follows a χ^2 distribution with 1 degree of freedom under the null hypothesis of conditional independence.

This method can be extended to general $I \times J \times K$ contingency tables, where I and J represent the number of rows and columns, respectively, and K is the number of strata.

Null Hypothesis (H_0):

There is no association between the two variables of interest across all strata, after controlling for the confounder.
In mathematical terms:

$$H_0 : \text{Odds Ratio (OR)} = 1 \text{ or Risk Ratio (RR)} = 1$$

Alternative Hypothesis (H_a):

There is an association between the two variables of interest across all strata, after controlling for the confounder.
In mathematical terms:

$$H_a : \text{Odds Ratio (OR)} \neq 1 \text{ or Risk Ratio (RR)} \neq 1$$

Let's consider a scenario where a business wants to evaluate the relationship between customer satisfaction (Satisfied vs. Not Satisfied) and the likelihood of repeat purchases (Yes vs. No) across different regions (e.g., North, South, and West). The goal is to determine whether this relationship holds consistently across the regions.

```
# Create a 2 x 2 x 3 contingency table
CustomerData = array(
    c(40, 30, 200, 300, 35, 20, 180, 265, 50, 25, 250, 275),
    dim = c(2, 2, 3),
    dimnames = list(
        Satisfaction = c("Satisfied", "Not Satisfied"),
        RepeatPurchase = c("Yes", "No"),
        Region = c("North", "South", "West")
    )
)

# View marginal table (summarized across regions)
margin.table(CustomerData, c(1, 2))
#>                RepeatPurchase
#> Satisfaction    Yes  No
#>   Satisfied     125 630
#>   Not Satisfied  75 840
```

1. Calculate the overall odds ratio (ignoring strata):

```
library(samplesizeCMH)
marginal_table = margin.table(CustomerData, c(1, 2))
odds.ratio(marginal_table)
#> [1] 2.222222
```

2. Calculate the conditional odds ratios for each region:

```
apply(CustomerData, 3, odds.ratio)
#>    North    South     West
#> 2.000000 2.576389 2.200000
```

3. The Mantel-Haenszel Test evaluates whether the relationship between customer satisfaction and repeat purchases remains consistent across regions:

```
mantelhaen.test(CustomerData, correct = TRUE)
#>
#>  Mantel-Haenszel chi-squared test with continuity correction
#>
#> data:  CustomerData
#> Mantel-Haenszel X-squared = 26.412, df = 1, p-value = 2.758e-07
#> alternative hypothesis: true common odds ratio is not equal to 1
#> 95 percent confidence interval:
#>  1.637116 3.014452
#> sample estimates:
#> common odds ratio
#>          2.221488
```

Interpretation

1. **Overall Odds Ratio**: This provides an estimate of the overall association between satisfaction and repeat purchases, ignoring regional differences.
2. **Conditional Odds Ratios**: These show whether the odds of repeat purchases given satisfaction are similar across regions.
3. **Mantel-Haenszel Test**: A significant test result (e.g., $p < 0.05$) suggests that the relationship between satisfaction and repeat purchases is consistent across regions. Conversely, a non-significant result implies that regional differences may affect the association. By applying the Mantel-Haenszel Test, businesses can determine if a marketing or customer retention strategy should be uniformly applied or customized to account for regional variations.
    1. There is strong evidence to suggest that the two variables of interest are associated across the strata (North, South, and West), even after accounting for potential confounding effects of stratification.
    2. The common odds ratio of approximately 2.22 indicates a substantial association, meaning that the outcome is more likely in the exposed group compared to the unexposed group.
    3. The variability in the stratum-specific odds ratios suggests that the strength of the association may differ slightly by region, but the Mantel-Haenszel test assumes the association is consistent (homogeneous).

4.5.2.2 McNemar's Test

McNemar's Test is a special case of the Mantel-Haenszel Chi-square Test, designed for paired nominal data. It is particularly useful for evaluating changes in categorical responses before and after a treatment or intervention, or for comparing paired responses in matched samples. Unlike the Mantel-Haenszel Test, which handles stratified data, McNemar's Test is tailored to situations with a single 2×2 table derived from paired observations.

McNemar's Test assesses whether the proportions of discordant pairs (off-diagonal elements in a 2×2 table) are significantly different. Specifically, it tests the null hypothesis that the probabilities of transitioning from one category to another are equal.

- **Null Hypothesis (H_0)**:

  $$P(\text{Switch from A to B}) = P(\text{Switch from B to A})$$

  This implies that the probabilities of transitioning from one category to the other are equal, or equivalently, the off-diagonal cell counts (n_{12} and n_{21}) are symmetric:

  $$H_0 : n_{12} = n_{21}$$

- **Alternative Hypothesis (H_A)**:

  $$P(\text{Switch from A to B}) \neq P(\text{Switch from B to A})$$

  This suggests that the probabilities of transitioning between categories are not equal, or equivalently, the off-diagonal cell counts (n_{12} and n_{21}) are asymmetric:

  $$H_A : n_{12} \neq n_{21}$$

For example, consider a business analyzing whether a new advertising campaign influences customer preference for two products (A and B). Each customer is surveyed before and after the campaign, resulting in the following 2×2 contingency table:

- **Before rows**: Preference for Product A or B before the campaign.
- **After columns**: Preference for Product A or B after the campaign.

Let the table structure be:

	After A	After B
Before A	n_{11}	n_{12}
Before B	n_{21}	n_{22}

- n_{12}: Customers who switched from Product A to B.
- n_{21}: Customers who switched from Product B to A.

The test focuses on n_{12} and n_{21}, as they represent the discordant pairs.

The McNemar's Test statistic is:

$$M^2 = \frac{(|n_{12} - n_{21}| - 0.5)^2}{n_{12} + n_{21}}$$

where

- The 0.5 is a continuity correction applied when sample sizes are small.
- Under the null hypothesis of no preference change, M^2 follows a χ^2 distribution with 1 degree of freedom.

Let's analyze a voting behavior study where participants were surveyed before and after a campaign. The table represents:

- Rows: Voting preference before the campaign (Yes, No).
- Columns: Voting preference after the campaign (Yes, No).

```r
# Voting preference before and after a campaign
vote = matrix(c(682, 22, 86, 810), nrow = 2, byrow = TRUE,
              dimnames = list(
                "Before" = c("Yes", "No"),
                "After" = c("Yes", "No")
              ))

# Perform McNemar's Test with continuity correction
mcnemar_result <- mcnemar.test(vote, correct = TRUE)
mcnemar_result
#>
#>  McNemar's Chi-squared test with continuity correction
#>
#> data:  vote
#> McNemar's chi-squared = 36.75, df = 1, p-value = 1.343e-09
```

The test provides:

- **Test statistic (M^2)**: Quantifies the asymmetry in discordant pairs.
- **p-value**: Indicates whether there is a significant difference in the discordant proportions.

Interpretation

1. **Test Statistic**: A large M^2 value suggests significant asymmetry in the discordant pairs.
2. **p-value**:

- A low p-value (e.g., $p < 0.05$) rejects the null hypothesis, indicating that the proportion of participants switching preferences (e.g., from Yes to No) is significantly different from those switching in the opposite direction (e.g., from No to Yes).
- A high p-value fails to reject the null hypothesis, suggesting no significant preference change.

McNemar's Test is widely used in business and other fields:

1. **Marketing Campaigns**: Evaluating whether a campaign shifts consumer preferences or purchase intentions.
2. **Product Testing**: Determining if a new feature or redesign changes customer ratings.
3. **Healthcare Studies**: Analyzing treatment effects in paired medical trials.

4.5.2.3 McNemar-Bowker Test

The McNemar-Bowker Test is an extension of McNemar's Test, designed for analyzing paired nominal data with more than two categories. It evaluates the symmetry of the full contingency table by comparing the off-diagonal elements across all categories. This test is particularly useful for understanding whether changes between categories are uniformly distributed or whether significant asymmetries exist.

Let the data be structured in an $r \times r$ square contingency table, where r is the number of categories, and the off-diagonal elements represent transitions between categories.

The hypotheses for the McNemar-Bowker Test are:

- **Null Hypothesis (H_0)**:

  $$P(\text{Switch from Category } i \text{ to Category } j) = P(\text{Switch from Category } j \text{ to Category } i) \quad \forall\, i \neq j$$

  This implies that the off-diagonal elements are symmetric, and there is no directional preference in category transitions.

- **Alternative Hypothesis (H_A)**:

  $$P(\text{Switch from Category } i \text{ to Category } j) \neq P(\text{Switch from Category } j \text{ to Category } i) \quad \text{for at leas}$$

  This suggests that the off-diagonal elements are not symmetric, indicating a directional preference in transitions between at least one pair of categories.

The McNemar-Bowker Test statistic is:

$$B^2 = \sum_{i<j} \frac{(n_{ij} - n_{ji})^2}{n_{ij} + n_{ji}}$$

where

- n_{ij}: Observed count of transitions from category i to category j.
- n_{ji}: Observed count of transitions from category j to category i.

Under the null hypothesis, the test statistic B^2 approximately follows a χ^2 distribution with $\frac{r(r-1)}{2}$ degrees of freedom (corresponding to the number of unique pairs of categories).

For example, a company surveys customers about their satisfaction before and after implementing a new policy. Satisfaction is rated on a scale of 1 to 3 (1 = Low, 2 = Medium, 3 = High). The paired responses are summarized in the following 3×3 contingency table.

```r
# Satisfaction ratings before and after the intervention
satisfaction_table <- matrix(c(
    30, 10, 5,  # Before: Low
    8, 50, 12,  # Before: Medium
    6, 10, 40   # Before: High
), nrow = 3, byrow = TRUE,
dimnames = list(
    "Before" = c("Low", "Medium", "High"),
    "After" = c("Low", "Medium", "High")
))

# Function to perform McNemar-Bowker Test
mcnemar_bowker_test <- function(table) {
  if (!all(dim(table)[1] == dim(table)[2])) {
    stop("Input must be a square matrix.")
  }

  # Extract off-diagonal elements
  n <- nrow(table)
  stat <- 0
  df <- 0

  for (i in 1:(n - 1)) {
    for (j in (i + 1):n) {
      nij <- table[i, j]
      nji <- table[j, i]
```

```
      stat <- stat + (nij - nji)^2 / (nij + nji)
      df <- df + 1
    }
  }

  p_value <- pchisq(stat, df = df, lower.tail = FALSE)
  return(list(statistic = stat, df = df, p_value = p_value))
}

# Run the test
result <- mcnemar_bowker_test(satisfaction_table)

# Print results
cat("McNemar-Bowker Test Results:\n")
#> McNemar-Bowker Test Results:
cat("Test Statistic (B^2):", result$statistic, "\n")
#> Test Statistic (B^2): 0.4949495
cat("Degrees of Freedom:", result$df, "\n")
#> Degrees of Freedom: 3
cat("p-value:", result$p_value, "\n")
#> p-value: 0.9199996
```

The output includes:

- **Test Statistic** (B^2): A measure of the asymmetry in the off-diagonal elements.
- **p-value**: The probability of observing the data under the null hypothesis of symmetry.

Interpretation

1. **Test Statistic**: A large B^2 value suggests substantial asymmetry in transitions between categories.
2. **p-value**:
   - If the **p-value is less than the significance level** (e.g., $p < 0.05$), we reject the null hypothesis, indicating significant asymmetry in the transitions between at least one pair of categories.
   - If the **p-value is greater than the significance level**, we fail to reject the null hypothesis, suggesting that the category transitions are symmetric.

The McNemar-Bowker Test has broad applications in business and other fields:

1. **Customer Feedback Analysis**: Evaluating changes in customer satisfaction levels before and after interventions.

2. **Marketing Campaigns**: Assessing shifts in brand preferences across multiple brands in response to an advertisement.
3. **Product Testing**: Understanding how user preferences among different product features change after a redesign.

4.5.2.4 Stuart-Maxwell Test

The Stuart-Maxwell Test is used for analyzing changes in paired categorical data with more than two categories. It is a generalization of McNemar's Test, applied to square contingency tables where the off-diagonal elements represent transitions between categories. Unlike the McNemar-Bowker Test, which tests for symmetry across all pairs, the Stuart-Maxwell Test focuses on overall marginal homogeneity.

The test evaluates whether the marginal distributions of paired data are consistent across categories. This is particularly useful when investigating whether the distribution of responses has shifted between two conditions, such as before and after an intervention.

Hypotheses for the Stuart-Maxwell Test

- **Null Hypothesis (H_0)**:

  The marginal distributions of the paired data are homogeneous (no difference).

- **Alternative Hypothesis (H_A)**:

  The marginal distributions of the paired data are not homogeneous (there is a difference).

The Stuart-Maxwell Test statistic is calculated as:

$$M^2 = \mathbf{b}'\mathbf{V}^{-1}\mathbf{b}$$

where:

- $\mathbf{b}$: Vector of differences between the marginal totals of paired categories.
- $\mathbf{V}$: Covariance matrix of $\mathbf{b}$ under the null hypothesis.

The test statistic M^2 follows a χ^2 distribution with $(r-1)$ degrees of freedom, where r is the number of categories.

A company surveys employees about their satisfaction levels (Low, Medium, High) before and after implementing a new workplace policy. The results are summarized in the following 3×3 contingency table.

```r
# Employee satisfaction data before and after a policy change
satisfaction_table <- matrix(c(
    40, 10, 5,  # Before: Low
    8, 50, 12,  # Before: Medium
    6, 10, 40   # Before: High
), nrow = 3, byrow = TRUE,
dimnames = list(
    "Before" = c("Low", "Medium", "High"),
    "After" = c("Low", "Medium", "High")
))

# Function to perform the Stuart-Maxwell Test
stuart_maxwell_test <- function(table) {
  if (!all(dim(table)[1] == dim(table)[2])) {
    stop("Input must be a square matrix.")
  }

  # Marginal totals for each category
  row_totals <- rowSums(table)
  col_totals <- colSums(table)

  # Vector of differences between row and column marginal totals
  b <- row_totals - col_totals

  # Covariance matrix under the null hypothesis
  total <- sum(table)
  V <- diag(row_totals + col_totals) -
       (outer(row_totals, col_totals, "+") / total)

  # Calculate the test statistic
  M2 <- t(b) %*% solve(V) %*% b
  df <- nrow(table) - 1
  p_value <- pchisq(M2, df = df, lower.tail = FALSE)

  return(list(statistic = M2, df = df, p_value = p_value))
}

# Run the Stuart-Maxwell Test
result <- stuart_maxwell_test(satisfaction_table)

# Print the results
cat("Stuart-Maxwell Test Results:\n")
#> Stuart-Maxwell Test Results:
cat("Test Statistic (M^2):", result$statistic, "\n")
#> Test Statistic (M^2): 0.01802387
```

```r
cat("Degrees of Freedom:", result$df, "\n")
#> Degrees of Freedom: 2
cat("p-value:", result$p_value, "\n")
#> p-value: 0.9910286
```

Interpretation

1. **Test Statistic**: Measures the extent of marginal differences in the table.
2. **p-value**:
   - A **low p-value** (e.g., $p < 0.05$) indicates significant differences between the marginal distributions, suggesting a change in the distribution of responses.
   - A **high p-value** suggests no evidence of marginal differences, meaning the distribution is consistent across conditions.

Practical Applications of the Stuart-Maxwell Test

1. **Employee Surveys**: Analyzing shifts in satisfaction levels before and after policy changes.
2. **Consumer Studies**: Evaluating changes in product preferences before and after a marketing campaign.
3. **Healthcare Research**: Assessing changes in patient responses to treatments across categories.

4.5.2.5 Cochran-Mantel-Haenszel (CMH) Test

The Cochran-Mantel-Haenszel (CMH) Test is a generalization of the Mantel-Haenszel Chi-square Test. It evaluates the association between two variables while controlling for the effect of a third stratifying variable. This test is particularly suited for ordinal data, allowing researchers to detect trends and associations across strata.

The CMH Test addresses scenarios where:

1. Two variables (e.g., exposure and outcome) are ordinal or nominal.
2. A third variable (e.g., a demographic or environmental factor) stratifies the data into K independent groups.

The test answers: Is there a consistent association between the two variables across the strata defined by the third variable?

The CMH Test has three main variations depending on the nature of the data:

1. **Correlation Test for Ordinal Data**: Assesses whether there is a linear association between two ordinal variables across strata.

2. **General Association Test**: Tests for any association (not necessarily ordinal) between two variables while stratifying by a third.
3. **Homogeneity Test**: Checks whether the strength of the association between the two variables is consistent across strata.

Hypotheses

- **Null Hypothesis (H_0)**: There is no association between the two variables across all strata, or the strength of the association is consistent across strata.
- **Alternative Hypothesis (H_A)**: There is an association between the two variables in at least one stratum, or the strength of the association varies across strata.

The CMH test statistic is:

$$CMH = \frac{\left(\sum_k (O_k - E_k)\right)^2}{\sum_k V_k}$$

Where:

- O_k: Observed counts in stratum k.
- E_k: Expected counts in stratum k, calculated under the null hypothesis.
- V_k: Variance of the observed counts in stratum k.

The test statistic follows a χ^2 distribution with 1 degree of freedom under the null hypothesis.

A company evaluates whether sales performance (Low, Medium, High) is associated with product satisfaction (Low, Medium, High) across three experience levels (Junior, Mid-level, Senior). The data is organized into a $3 \times 3 \times 3$ contingency table.

```r
# Sales performance data
sales_data <- array(
    c(20, 15, 10, 12, 18, 15, 8, 12, 20,   # Junior
      25, 20, 15, 20, 25, 30, 10, 15, 20,  # Mid-level
      30, 25, 20, 28, 32, 35, 15, 20, 30), # Senior
    dim = c(3, 3, 3),
    dimnames = list(
        SalesPerformance = c("Low", "Medium", "High"),
        Satisfaction = c("Low", "Medium", "High"),
        ExperienceLevel = c("Junior", "Mid-level", "Senior")
```

```
    )
)

# Load the vcd package for the CMH test
library(vcd)

# Perform CMH Test
cmh_result <- mantelhaen.test(sales_data, correct = FALSE)
cmh_result
#>
#>  Cochran-Mantel-Haenszel test
#>
#> data:  sales_data
#> Cochran-Mantel-Haenszel M^2 = 22.454, df = 4, p-value = 0.0001627
```

Interpretation

1. **Test Statistic**: A large CMH statistic suggests a significant association between sales performance and satisfaction after accounting for experience level.
2. **p-value**:
   - A **low p-value** (e.g., $p < 0.05$) indicates a significant association between the two variables across strata.
   - A **high p-value** suggests no evidence of association or that the relationship is consistent across all strata.

Practical Applications of the CMH Test

1. **Business Performance Analysis**: Investigating the relationship between customer satisfaction and sales performance across different demographic groups.
2. **Healthcare Studies**: Assessing the effect of treatment (e.g., dosage) on outcomes while controlling for patient characteristics (e.g., age groups).
3. **Educational Research**: Analyzing the relationship between test scores and study hours, stratified by teaching method.

4.5.2.6 Summary Table of Tests

The following table provides a concise guide on when and why to use each test:

Test Name	When to Use	Key Question Addressed	Data Requirements
[**Mantel-Haenszel Chi-square Test**]	When testing for association between two binary variables across multiple strata.	Is there a consistent association across strata?	Binary variables in $2 \times 2 \times K$ tables.
[**McNemar's Test**]	When analyzing marginal symmetry in paired binary data.	Are the proportions of discordant pairs equal?	Paired binary responses (2×2 table).
[**McNemar-Bowker Test**]	When testing for symmetry in paired nominal data with more than two categories.	Are the off-diagonal elements symmetric across all categories?	Paired nominal data in $r \times r$ tables.
[**Cochran-Mantel-Haenszel (CMH) Test**]	When testing ordinal or general associations while controlling for a stratifying variable.	Is there an association between two variables after stratification?	Ordinal or nominal data in $I \times J \times K$ tables.
[**Stuart-Maxwell Test**]	When analyzing marginal homogeneity in paired nominal data with more than two categories.	Are the marginal distributions of paired data homogeneous?	Paired nominal data in $r \times r$ tables.

How to Choose the Right Test

1. **Paired vs. Stratified Data**:
   - Use **McNemar's Test** or **McNemar-Bowker Test** for paired data.
   - Use **Mantel-Haenszel** or **CMH Test** for stratified data.
2. **Binary vs. Multi-category Variables**:
   - Use **McNemar's Test** for binary data.
   - Use **McNemar-Bowker Test** or **Stuart-Maxwell Test** for multi-category data.
3. **Ordinal Trends**:
   - Use the **CMH Test** if testing for ordinal associations while controlling for a stratifying variable.

4.5.3 Ordinal Trend

When analyzing ordinal data, it is often important to determine whether a consistent trend exists between variables. Tests for trend are specifically designed to detect monotonic relationships where changes in one variable are systematically associated with changes in another. These tests are widely used in scenarios involving ordered categories, such as customer satisfaction ratings, income brackets, or educational levels.

The primary objectives of trend tests are:

1. **To detect monotonic relationships**: Determine if higher or lower categories of one variable are associated with higher or lower categories of another variable.
2. **To account for ordinal structure**: Leverage the inherent order in the data to provide more sensitive and interpretable results compared to tests designed for nominal data.

Key Considerations for Trend Tests

1. **Data Structure**:
   - Ensure that the variables have a natural order and are treated as ordinal.
   - Verify that the trend test chosen matches the data structure (e.g., binary outcome vs. multi-level ordinal variables).
2. **Assumptions**:
   - Many tests assume monotonic trends, meaning that the relationship should not reverse direction.
3. **Interpretation**:
   - A significant result indicates the presence of a trend but does not imply causality.
   - The direction and strength of the trend should be carefully interpreted in the context of the data.

4.5.3.1 Cochran-Armitage Test

The Cochran-Armitage Test for Trend is a statistical method designed to detect a linear trend in proportions across ordered categories of a predictor variable. It is particularly useful in $2 \times J$ contingency tables, where there is a binary outcome (e.g., success/failure) and an ordinal predictor variable with J ordered levels.

The Cochran-Armitage Test evaluates whether the proportion of a binary outcome changes systematically across the levels of an ordinal predictor. This test leverages the ordinal nature of the predictor to enhance sensitivity and power compared to general chi-square tests.

Hypotheses

- **Null Hypothesis (H_0)**:

  The proportion of the binary outcome is constant across the levels of the ordinal predictor.

- **Alternative Hypothesis (H_A)**:

  There is a linear trend in the proportion of the binary outcome across the levels of the ordinal pre

---

The Cochran-Armitage Test statistic is calculated as:

$$Z = \frac{\sum_{j=1}^{J} w_j(n_{1j} - N_j\hat{p})}{\sqrt{\hat{p}(1-\hat{p})\sum_{j=1}^{J} w_j^2 N_j}}$$

Where:

- n_{1j}: Count of the binary outcome (e.g., "success") in category j.
- N_j: Total number of observations in category j.
- $\hat{p}$: Overall proportion of the binary outcome, calculated as:

$$\hat{p} = \frac{\sum_{j=1}^{J} n_{1j}}{\sum_{j=1}^{J} N_j}$$

- w_j: Score assigned to the jth category of the ordinal predictor, often set to j for equally spaced levels.

The test statistic Z follows a standard normal distribution under the null hypothesis.

Key Assumptions

1. **Ordinal Predictor**: The categories of the predictor variable must have a natural order.
2. **Binary Outcome**: The response variable must be dichotomous (e.g., success/failure).
3. **Independent Observations**: Observations within and across categories are independent.

Let's consider a study examining whether the success rate of a marketing campaign varies across three income levels (Low, Medium, High). The data is structured in a 2×3 contingency table:

Income Level	Success	Failure	Total
Low	20	30	50
Medium	35	15	50
High	45	5	50

```r
# Data: Success and Failure counts by Income Level
income_levels <- c("Low", "Medium", "High")
success <- c(20, 35, 45)
failure <- c(30, 15, 5)
total <- success + failure

# Scores for ordinal levels (can be custom weights)
scores <- 1:length(income_levels)

# Cochran-Armitage Test
# Function to calculate Z statistic
cochran_armitage_test <- function(success, failure, scores) {
  N <- success + failure
  p_hat <- sum(success) / sum(N)
  weights <- scores

  # Calculate numerator
  numerator <- sum(weights * (success - N * p_hat))

  # Calculate denominator
  denominator <- sqrt(p_hat * (1 - p_hat) * sum(weights^2 * N))

  # Z statistic
  Z <- numerator / denominator
  p_value <- 2 * (1 - pnorm(abs(Z)))

  return(list(Z_statistic = Z, p_value = p_value))
}

# Perform the test
result <- cochran_armitage_test(success, failure, scores)

# Print results
cat("Cochran-Armitage Test for Trend Results:\n")
#> Cochran-Armitage Test for Trend Results:
cat("Z Statistic:", result$Z_statistic, "\n")
#> Z Statistic: 2.004459
cat("p-value:", result$p_value, "\n")
#> p-value: 0.04502088
```

Interpretation

1. **Test Statistic (Z)**:
   - The Z value indicates the strength and direction of the trend.
   - Positive Z: Proportions increase with higher categories.
   - Negative Z: Proportions decrease with higher categories.
2. **p-value**:
   - A **low p-value** (e.g., $p < 0.05$) rejects the null hypothesis, indicating a significant linear trend.
   - A **high p-value** fails to reject the null hypothesis, suggesting no evidence of a trend.

Practical Applications

1. **Marketing**: Analyzing whether customer success rates vary systematically across income levels or demographics.
2. **Healthcare**: Evaluating the dose-response relationship between medication levels and recovery rates.
3. **Education**: Studying whether pass rates improve with higher levels of educational support.

4.5.3.2 Jonckheere-Terpstra Test

The Jonckheere-Terpstra Test is a nonparametric test designed to detect ordered differences between groups. It is particularly suited for ordinal data where both the predictor and response variables exhibit a monotonic trend. Unlike general nonparametric tests like the Kruskal-Wallis test, which assess any differences between groups, the Jonckheere-Terpstra Test specifically evaluates whether the data follows a prespecified ordering.

The Jonckheere-Terpstra Test determines whether:

1. There is a monotonic trend in the response variable across ordered groups of the predictor.
2. The data aligns with an a priori hypothesized order (e.g., group 1 < group 2 < group 3).

Hypotheses

- **Null Hypothesis (H_0)**:

  There is no trend in the response variable across the ordered groups.

- **Alternative Hypothesis (H_A)**:

  The response variable exhibits a monotonic trend across the ordered groups.

The trend can be increasing, decreasing, or as otherwise hypothesized.

The Jonckheere-Terpstra Test statistic is based on the number of pairwise comparisons (U) that are consistent with the hypothesized trend. For k groups:

1. Compare all possible pairs of observations across groups.
2. Count the number of pairs where the values are consistent with the hypothesized order.

The test statistic T is the sum of all pairwise comparisons:

$$T = \sum_{i<j} T_{ij}$$

Where T_{ij} is the number of concordant pairs between groups i and j.

Under the null hypothesis, T follows a normal distribution with:

- Mean:
$$\mu_T = \frac{N(N-1)}{4}$$
- Variance:
$$\sigma_T^2 = \frac{N(N-1)(2N+1)}{24}$$
  Where N is the total number of observations.

The standardized test statistic is:

$$Z = \frac{T - \mu_T}{\sigma_T}$$

Key Assumptions

1. **Ordinal or Interval Data**: The response variable must be at least ordinal, and the groups must have a logical order.
2. **Independent Groups**: Observations within and between groups are independent.
3. **Consistent Hypothesis**: The trend (e.g., increasing or decreasing) must be specified in advance.

Let's consider a study analyzing whether customer satisfaction ratings (on a scale of 1 to 5) improve with increasing levels of service tiers (Basic, Standard, Premium). The data is grouped by service tier, and we hypothesize that satisfaction ratings increase with higher service tiers.

```r
# Example Data: Customer Satisfaction Ratings by Service Tier
satisfaction <- list(
  Basic = c(2, 3, 2, 4, 3),
  Standard = c(3, 4, 3, 5, 4),
  Premium = c(4, 5, 4, 5, 5)
)

# Prepare data
ratings <- unlist(satisfaction)
groups <- factor(rep(names(satisfaction), times = sapply(satisfaction, length)))

# Calculate pairwise comparisons
manual_jonckheere <- function(ratings, groups) {
  n_groups <- length(unique(groups))
  pairwise_comparisons <- 0
  total_pairs <- 0

  # Iterate over group pairs
  for (i in 1:(n_groups - 1)) {
    for (j in (i + 1):n_groups) {
      group_i <- ratings[groups == levels(groups)[i]]
      group_j <- ratings[groups == levels(groups)[j]]

      # Count concordant pairs
      for (x in group_i) {
        for (y in group_j) {
          if (x < y) pairwise_comparisons <- pairwise_comparisons + 1
          if (x == y) pairwise_comparisons <- pairwise_comparisons + 0.5
          total_pairs <- total_pairs + 1
        }
      }
    }
  }

  # Compute test statistic
  T <- pairwise_comparisons
  N <- length(ratings)
  mu_T <- total_pairs / 2
  sigma_T <- sqrt(total_pairs * (N + 1) / 12)

  Z <- (T - mu_T) / sigma_T
  p_value <- 2 * (1 - pnorm(abs(Z)))

  return(list(T_statistic = T, Z_statistic = Z, p_value = p_value))
}
```

```r
# Perform the test
result <- manual_jonckheere(ratings, groups)

# Print results
cat("Jonckheere-Terpstra Test Results:\n")
#> Jonckheere-Terpstra Test Results:
cat("T Statistic (Sum of Concordant Pairs):", result$T_statistic, "\n")
#> T Statistic (Sum of Concordant Pairs): 49.5
cat("Z Statistic:", result$Z_statistic, "\n")
#> Z Statistic: 1.2
cat("p-value:", result$p_value, "\n")
#> p-value: 0.2301393
```

Interpretation

1. **Test Statistic (T)**:
   - Represents the sum of all pairwise comparisons consistent with the hypothesized order.
   - Includes 0.5 for tied pairs.
2. **Z Statistic**:
   - A standardized measure of the strength of the trend.
   - Calculated using T, the expected value of T under the null hypothesis (μ_T), and the variance of T (σ_T^2).
3. **p-value**:
   - A **low p-value** (e.g., $p < 0.05$) rejects the null hypothesis, indicating a significant trend in the response variable across ordered groups.
   - A **high p-value** fails to reject the null hypothesis, suggesting no evidence of a trend.

Practical Applications

1. **Customer Experience Analysis**: Assessing whether customer satisfaction increases with higher service levels or product tiers.
2. **Healthcare Studies**: Testing whether recovery rates improve with increasing doses of a treatment.
3. **Education Research**: Analyzing whether test scores improve with higher levels of educational intervention.

4.5.3.3 Mantel Test for Trend

The Mantel Test for Trend is a statistical method designed to detect a linear association between two ordinal variables. It is an extension of the Mantel-Haenszel

Chi-square Test and is particularly suited for analyzing trends in ordinal contingency tables, such as $I \times J$ tables where both variables are ordinal.

The Mantel Test for Trend evaluates whether an increasing or decreasing trend exists between two ordinal variables. It uses the ordering of categories to assess linear relationships, making it more sensitive to trends compared to general association tests like chi-square.

Hypotheses

- **Null Hypothesis (H_0)**:

  There is no linear association between the two ordinal variables.

- **Alternative Hypothesis (H_A)**:

  There is a significant linear association between the two ordinal variables.

The Mantel Test is based on the Pearson correlation between the row and column scores in an ordinal contingency table. The test statistic is:

$$M = \frac{\sum_i \sum_j w_i w_j n_{ij}}{\sqrt{\sum_i w_i^2 n_{i\cdot} \sum_j w_j^2 n_{\cdot j}}}$$

Where:

- n_{ij}: Observed frequency in cell (i, j).
- $n_{i\cdot}$: Row marginal total for row i.
- $n_{\cdot j}$: Column marginal total for column j.
- w_i: Score for the ith row.
- ore for the jth column.

The test statistic M is asymptotically normally distributed under the null hypothesis.

Key Assumptions

1. **Ordinal Variables**: Both variables must have a natural order.
2. **Linear Trend**: Assumes a linear relationship between the scores assigned to the rows and columns.
3. **Independence**: Observations must be independent.

Let's consider a marketing study evaluating whether customer satisfaction levels (Low, Medium, High) are associated with increasing purchase frequency (Low, Medium, High).

```
# Customer satisfaction and purchase frequency data
data <- matrix(
  c(10, 5, 2, 15, 20, 8, 25, 30, 12),
  nrow = 3,
  byrow = TRUE,
  dimnames = list(
    Satisfaction = c("Low", "Medium", "High"),
    Frequency = c("Low", "Medium", "High")
  )
)

# Assign scores for rows and columns
row_scores <- 1:nrow(data)
col_scores <- 1:ncol(data)

# Compute Mantel statistic manually
mantel_test_manual <- function(data, row_scores, col_scores) {
  numerator <- sum(outer(row_scores, col_scores, "*") * data)
  row_marginals <- rowSums(data)
  col_marginals <- colSums(data)
  row_variance <- sum(row_scores^2 * row_marginals)
  col_variance <- sum(col_scores^2 * col_marginals)

  M <- numerator / sqrt(row_variance * col_variance)
  z_value <- M
  p_value <- 2 * (1 - pnorm(abs(z_value))) # Two-tailed test

  return(list(Mantel_statistic = M, p_value = p_value))
}

# Perform the Mantel Test
result <- mantel_test_manual(data, row_scores, col_scores)

# Display results
cat("Mantel Test for Trend Results:\n")
#> Mantel Test for Trend Results:
cat("Mantel Statistic (M):", result$Mantel_statistic, "\n")
#> Mantel Statistic (M): 0.8984663
cat("p-value:", result$p_value, "\n")
#> p-value: 0.368937
```

Interpretation

1. **Test Statistic (M)**:
   - Represents the strength and direction of the linear association.
   - Positive M: Increasing trend.
   - Negative M: Decreasing trend.
2. **p-value**:
   - A **low p-value** (e.g., $p < 0.05$) indicates a significant linear association.
   - A **high p-value** suggests no evidence of a trend.

Practical Applications

1. **Marketing Analysis**: Investigating whether satisfaction levels are associated with purchase behavior or loyalty.
2. **Healthcare Research**: Testing for a dose-response relationship between treatment levels and outcomes.
3. **Social Sciences**: Analyzing trends in survey responses across ordered categories.

4.5.3.4 Chi-square Test for Linear Trend

The Chi-square Test for Linear Trend is a statistical method used to detect a linear relationship between an ordinal predictor and a binary outcome. It is an extension of the chi-square test, designed specifically for ordered categories, making it more sensitive to linear trends in proportions compared to a general chi-square test of independence.

The Chi-square Test for Linear Trend evaluates whether the proportions of a binary outcome (e.g., success/failure) change systematically across ordered categories of a predictor variable. It is widely used in situations such as analyzing dose-response relationships or evaluating trends in survey responses.

Hypotheses

- **Null Hypothesis (H_0)**:

  There is no linear trend in the proportions of the binary outcome across ordered categories.

- **Alternative Hypothesis (H_A)**:

  There is a significant linear trend in the proportions of the binary outcome across ordered categor

---

The test statistic is:

$$X^2_{\text{trend}} = \frac{\left(\sum_{j=1}^{J} w_j(p_j - \bar{p})N_j\right)^2}{\sum_{j=1}^{J} w_j^2 \bar{p}(1-\bar{p})N_j}$$

Where: - J: Number of ordered categories. - w_j: Scores assigned to the jth category (typically $j = 1, 2, \ldots, J$). - p_j: Proportion of success in the jth category. - $\bar{p}$: Overall proportion of success across all categories. - N_j: Total number of observations in the jth category.

The test statistic follows a chi-square distribution with 1 degree of freedom under the null hypothesis.

Key Assumptions

1. **Binary Outcome**: The response variable must be binary (e.g., success/failure).
2. **Ordinal Predictor**: The predictor variable must have a natural order.
3. **Independent Observations**: Data across categories must be independent.

Let's consider a study analyzing whether the proportion of customers who recommend a product increases with customer satisfaction levels (Low, Medium, High).

```r
# Example Data: Customer Satisfaction and Recommendation
satisfaction_levels <- c("Low", "Medium", "High")
success <- c(20, 35, 50)  # Number of customers who recommend the product
failure <- c(30, 15, 10)  # Number of customers who do not recommend the product
total <- success + failure

# Assign ordinal scores
scores <- 1:length(satisfaction_levels)

# Calculate overall proportion of success
p_hat <- sum(success) / sum(total)
```

Interpretation

1. **Chi-square Statistic (X^2_{trend})**:
    - Indicates the strength of the linear trend in the proportions.
2. **p-value**:

- A **low p-value** (e.g., $p < 0.05$) rejects the null hypothesis, indicating a significant linear trend.
- A **high p-value** suggests no evidence of a linear trend.

Practical Applications

1. **Marketing**: Analyzing whether customer satisfaction levels predict product recommendations or repurchase intentions.
2. **Healthcare**: Evaluating dose-response relationships in clinical trials.
3. **Education**: Testing whether higher levels of intervention improve success rates.

4.5.3.5 Key Takeways

Test	Purpose	Key Assumptions	Use Cases
[**Cochran-Armitage Test**]	Tests for a linear trend in proportions across ordinal categories.	- Binary response variable. - Predictor variable is ordinal.	Evaluating dose-response relationships, comparing proportions across ordinal groups.
[**Jonckheere-Terpstra Test**]	Tests for a monotonic trend in a response variable across ordered groups.	- Response variable is continuous or ordinal. - Predictor variable is ordinal.	Comparing medians or distributions across ordinal groups, e.g., treatment levels.
[**Mantel Test for Trend**]	Evaluates a linear association between an ordinal predictor and response.	- Ordinal variables. - Linear trend expected.	Determining trends in stratified or grouped data.
[**Chi-square Test for Linear Trend**]	Tests for linear trends in categorical data using contingency tables.	- Contingency table with ordinal predictor. - Sufficient sample size (expected frequencies > 5).	Analyzing trends in frequency data, e.g., examining disease prevalence by age groups.

4.6 Divergence Metrics and Tests for Comparing Distributions

Divergence metrics are powerful tools used to measure the similarity or dissimilarity between probability distributions. Unlike deviation and deviance statistics, divergence metrics focus on the broader relationships between entire distributions, rather than individual data points or specific model fit metrics. Let's clarify these differences:

- **Deviation Statistics**: Measure the difference between the realization of a variable and some reference value (e.g., the mean). Common statistics derived from deviations include:
  - Standard deviation
  - Average absolute deviation
  - Median absolute deviation
  - Maximum absolute deviation
- **Deviance Statistics**: Assess the goodness-of-fit of statistical models. These are analogous to the sum of squared residuals in ordinary least squares (OLS) but are generalized for use in cases with maximum likelihood estimation (MLE). Deviance statistics are frequently employed in generalized linear models (GLMs).

Divergence statistics differ fundamentally by focusing on statistical distances between entire probability distributions, rather than on individual data points or model errors.

1. Divergence Metrics

- **Definition**: Divergence metrics measure how much one probability distribution differs from another.
- **Key Properties**:
  - **Asymmetry**: Many divergence metrics, such as Kullback-Leibler (KL) divergence, are not symmetric (i.e., $D(P\|\|Q) \neq D(Q\|\|P)$).
  - **Non-Metric**: They don't necessarily satisfy the properties of a metric (e.g., symmetry, triangle inequality).
  - **Unitless**: Divergences are often expressed in terms of information (e.g., bits or nats).
- **When to Use**:
  - Use divergence metrics to assess the degree of mismatch between two probability distributions, especially in machine learning, statistical inference, or model evaluation.

2. Distance Metrics

- **Definition**: Distance metrics measure the "distance" or dissimilarity between two objects, including probability distributions, datasets, or points in space.
- **Key Properties**:
  - **Symmetry**: $D(P, Q) = D(Q, P)$.
  - **Triangle Inequality**: $D(P, R) \leq D(P, Q) + D(Q, R)$.
  - **Non-Negativity**: $D(P, Q) \geq 0$, with $D(P, Q) = 0$ only if $P = Q$.
- **When to Use**:
  - Use distance metrics to compare datasets, distributions, or clustering outcomes where symmetry and geometric properties are important.

Aspect	Divergence Metrics	Distance Metrics
Symmetry	Often asymmetric (e.g., KL divergence).	Always symmetric (e.g., Wasserstein).
Triangle Inequality	Not satisfied.	Satisfied.
Use Case	Quantifying how different distributions are.	Measuring the dissimilarity or "cost" of transformation.

Applications of Divergence Metrics

Divergence metrics have found wide utility across domains, including:

- **Detecting Data Drift in Machine Learning**: Used to monitor whether the distribution of incoming data differs significantly from training data.
- **Feature Selection**: Employed to identify features with the most distinguishing power by comparing their distributions across different classes.
- **Variational Autoencoders (VAEs)**: Divergence metrics (such as Kullback-Leibler divergence) are central to the loss functions used in training VAEs.
- **Reinforcement Learning**: Measure the similarity between policy distributions to improve decision-making processes.
- **Assessing Consistency**: Compare the distributions of two variables representing constructs to test their relationship or agreement.

Divergence metrics are also highly relevant in business settings, providing insights and solutions for a variety of applications, such as:

- **Customer Segmentation and Targeting**: Compare the distributions of customer demographics or purchase behavior across market segments to identify key differences and target strategies more effectively.

- **Market Basket Analysis**: Measure divergence between distributions of product co-purchases across regions or customer groups to optimize product bundling and cross-selling strategies.
- **Marketing Campaign Effectiveness**: Evaluate whether the distribution of customer responses (e.g., click-through rates or conversions) differs significantly before and after a marketing campaign, providing insights into its success.
- **Fraud Detection**: Monitor divergence in transaction patterns over time to detect anomalies that may indicate fraudulent activities.
- **Supply Chain Optimization**: Compare demand distributions across time periods or regions to optimize inventory allocation and reduce stockouts or overstocking.
- **Pricing Strategy Evaluation**: Analyze the divergence between pricing and purchase distributions across products or customer segments to refine pricing models and improve profitability.
- **Churn Prediction**: Compare distributions of engagement metrics (e.g., frequency of transactions or usage time) between customers likely to churn and those who stay, to design retention strategies.
- **Financial Portfolio Analysis**: Assess divergence between the expected returns and actual performance distributions of different asset classes to adjust investment strategies.

4.6.1 Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov (KS) test is a **non-parametric test** used to determine whether two distributions differ significantly or whether a sample distribution matches a reference distribution. It is applicable to **continuous distributions** and is widely used in hypothesis testing and model evaluation.

Mathematical Definition

The KS statistic is defined as:

$$D = \max |F_P(x) - F_Q(x)|$$

Where:

- $F_P(x)$ is the cumulative distribution function (CDF) of the first distribution (or sample).

- $F_Q(x)$ is the CDF of the second distribution (or theoretical reference distribution).
- D measures the maximum vertical distance between the two CDFs.

Hypotheses

- **Null Hypothesis** (H_0): The empirical distribution follows a specified distribution (or the two samples are drawn from the same distribution).
- **Alternative Hypothesis** (H_1): The empirical distribution does not follow the specified distribution (or the two samples are drawn from different distributions).

Properties of the KS Statistic

- **Range**:
$$D \in [0, 1]$$
  - $D = 0$: Perfect match between the distributions.
  - $D = 1$: Maximum dissimilarity between the distributions.
- **Non-parametric Nature**: The KS test makes no assumptions about the underlying distribution of the data.

The KS test is useful in various scenarios, including:

1. Comparing two empirical distributions to evaluate similarity.
2. Testing goodness-of-fit for a sample against a theoretical distribution.
3. Detecting data drift or shifts in distributions over time.
4. Validating simulation outputs by comparing them to real-world data.

Example 1: Continuous Distributions

```r
# Load necessary libraries
library(stats)

# Generate two sample distributions
set.seed(1)
sample_1 <- rnorm(100)        # Sample from a standard normal distribution
sample_2 <- rnorm(100, mean = 1)  # Sample with mean shifted to 1

# Perform Kolmogorov-Smirnov test
ks_test_result <- ks.test(sample_1, sample_2)
print(ks_test_result)
#>
#>  Asymptotic two-sample Kolmogorov-Smirnov test
#>
#> data:  sample_1 and sample_2
```

```
#> D = 0.36, p-value = 4.705e-06
#> alternative hypothesis: two-sided
```

This compares the CDFs of the two samples. The p-value indicates whether the null hypothesis (that the samples come from the same distribution) can be rejected.

Example 2: Discrete Data with Bootstrapped KS Test

For discrete data, a bootstrapped version of the KS test is often used to bypass the continuity requirement.

```
library(Matching)

# Define two discrete samples
discrete_sample_1 <- c(0:10)
discrete_sample_2 <- c(0:10)

# Perform bootstrapped KS test
ks_boot_result <- ks.boot(Tr = discrete_sample_1, Co = discrete_sample_2)
print(ks_boot_result)
#> $ks.boot.pvalue
#> [1] 1
#>
#> $ks
#>
#>  Exact two-sample Kolmogorov-Smirnov test
#>
#> data:  Tr and Co
#> D = 0, p-value = 1
#> alternative hypothesis: two-sided
#>
#>
#> $nboots
#> [1] 1000
#>
#> attr(,"class")
#> [1] "ks.boot"
```

This method performs a bootstrapped version of the KS test, suitable for discrete data. The p-value indicates whether the null hypothesis (that the samples come from the same distribution) can be rejected.

Example 3: Comparing Multiple Distributions with KL Divergence (Optional Enhancement)

If you wish to extend the analysis to include divergence measures like KL divergence, use the following:

```r
library(entropy)
library(tidyverse)

# Define multiple samples
lst <- list(sample_1 = c(1:20), sample_2 = c(2:30), sample_3 = c(3:30))

# Compute KL divergence between all pairs of distributions
result <- expand.grid(1:length(lst), 1:length(lst)) %>%
    rowwise() %>%
    mutate(KL = KL.empirical(lst[[Var1]], lst[[Var2]]))

print(result)
#> # A tibble: 9 x 3
#> # Rowwise:
#>    Var1  Var2     KL
#>   <int> <int>  <dbl>
#> 1     1     1 0
#> 2     2     1 0.150
#> 3     3     1 0.183
#> 4     1     2 0.704
#> 5     2     2 0
#> 6     3     2 0.0679
#> 7     1     3 0.622
#> 8     2     3 0.0870
#> 9     3     3 0
```

This calculates the KL divergence for all pairs of distributions in the list, offering additional insights into the relationships between the distributions.

4.6.2 Anderson-Darling Test

The Anderson-Darling (AD) test is a **goodness-of-fit test** that evaluates whether a sample of data comes from a specific distribution. It is an enhancement of the Kolmogorov-Smirnov test, with greater sensitivity to deviations in the tails of the distribution.

The Anderson-Darling test statistic is defined as:

$$A^2 = -n - \frac{1}{n}\sum_{i=1}^{n}\left[(2i-1)\left(\log F(Y_i) + \log(1 - F(Y_{n+1-i}))\right)\right]$$

Where:

- n is the sample size.

- F is the cumulative distribution function (CDF) of the theoretical distribution being tested.
- Y_i are the ordered sample values.

The AD test modifies the basic framework of the KS test by giving more weight to the tails of the distribution, making it particularly sensitive to tail discrepancies.

Hypotheses

- **Null Hypothesis** (H_0): The sample data follows the specified distribution.
- **Alternative Hypothesis** (H_1): The sample data does not follow the specified distribution.

Key Properties

1. **Tail Sensitivity**: Unlike the Kolmogorov-Smirnov test, the Anderson-Darling test emphasizes discrepancies in the tails of the distribution.
2. **Distribution-Specific Critical Values**: The AD test provides critical values tailored to the specific distribution being tested (e.g., normal, exponential).

The Anderson-Darling test is commonly used in:

1. Testing goodness-of-fit for a sample against theoretical distributions such as normal, exponential, or uniform.
2. Evaluating the appropriateness of parametric models in hypothesis testing.
3. Assessing distributional assumptions in quality control and reliability analysis.

Example: Testing Normality with the Anderson-Darling Test

```r
library(nortest)

# Generate a sample from a normal distribution
set.seed(1)
sample_data <- rnorm(100, mean = 0, sd = 1)

# Perform the Anderson-Darling test for normality
ad_test_result <- ad.test(sample_data)
print(ad_test_result)
#>
#>  Anderson-Darling normality test
#>
```

```
#> data:  sample_data
#> A = 0.16021, p-value = 0.9471
```

If the p-value is below a chosen significance level (e.g., 0.05), the null hypothesis that the data is normally distributed is rejected.

Example: Comparing Two Empirical Distributions

The AD test can also be applied to compare two empirical distributions using resampling techniques.

```
# Define two samples
set.seed(1)
sample_1 <- rnorm(100, mean = 0, sd = 1)
sample_2 <- rnorm(100, mean = 1, sd = 1)

# Perform resampling-based Anderson-Darling test (custom implementation or packages li
library(twosamples)
ad_test_result_empirical <- ad_test(sample_1, sample_2)
print(ad_test_result_empirical)
#>  Test Stat    P-Value
#> 6796.70454    0.00025
```

This evaluates whether the two empirical distributions differ significantly.

4.6.3 Chi-Square Goodness-of-Fit Test

The **Chi-Square Goodness-of-Fit Test** is a non-parametric statistical test used to evaluate whether a sample data set comes from a population with a specific distribution. It compares observed frequencies with expected frequencies under a hypothesized distribution.

- **Null Hypothesis** (H_0): The data follow the specified distribution.
- **Alternative Hypothesis** (H_a): The data do not follow the specified distribution.

The Chi-Square test statistic is computed as:

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

Where:

- O_i: Observed frequency for category i.
- E_i: Expected frequency for category i.
- k: Number of categories.

The test statistic follows a Chi-Square distribution with degrees of freedom:

$$\nu = k - 1 - p$$

Where p is the number of parameters estimated from the data.

Assumptions of the Test

1. **Random Sampling**: The sample data are drawn randomly from the population.
2. **Minimum Expected Frequency**: The expected frequencies E_i are sufficiently large (typically $E_i \geq 5$).
3. **Independence**: Observations in the sample are independent of each other.

Decision Rule

1. Compute the test statistic χ^2 using the observed and expected frequencies.
2. Determine the critical value $\chi^2_{\alpha,\nu}$ for the chosen significance level α and degrees of freedom ν.
3. Compare χ^2 to $\chi^2_{\alpha,\nu}$:
   - Reject H_0 if $\chi^2 > \chi^2_{\alpha,\nu}$.
   - Alternatively, use the p-value approach:
     - Reject H_0 if $p \leq \alpha$.
     - Fail to reject H_0 if $p > \alpha$.

Steps for the Chi-Square Goodness-of-Fit Test

1. Define the expected frequencies based on the hypothesized distribution.
2. Compute the observed frequencies from the data.
3. Calculate the test statistic χ^2.
4. Determine the degrees of freedom ν.
5. Compare χ^2 with the critical value or use the p-value for decision-making.

Example: Testing a Fair Die

Suppose you are testing whether a six-sided die is fair. The die is rolled 60 times, and the observed frequencies of the outcomes are:

- **Observed Frequencies**: $[10, 12, 8, 11, 9, 10]$
- **Expected Frequencies**: A fair die has equal probability for each face, so $E_i = 60/6 = 10$ for each face.

```r
# Observed frequencies
observed <- c(10, 12, 8, 11, 9, 10)

# Expected frequencies under a fair die
expected <- rep(10, 6)

# Perform Chi-Square Goodness-of-Fit Test
chisq_test <- chisq.test(x = observed, p = expected / sum(expected))

# Display results
chisq_test
#>
#>  Chi-squared test for given probabilities
#>
#> data:  observed
#> X-squared = 1, df = 5, p-value = 0.9626
```

Example: Testing a Loaded Die

For a die with unequal probabilities (e.g., a loaded die), the expected probabilities are defined explicitly:

```r
# Observed frequencies
observed <- c(10, 12, 8, 11, 9, 10)

# Expected probabilities (e.g., for a loaded die)
probabilities <- c(0.1, 0.2, 0.3, 0.1, 0.2, 0.1)

# Expected frequencies
expected <- probabilities * sum(observed)

# Perform Chi-Square Goodness-of-Fit Test
chisq_test_loaded <- chisq.test(x = observed, p = probabilities)

# Display results
chisq_test_loaded
#>
#>  Chi-squared test for given probabilities
#>
#> data:  observed
#> X-squared = 15.806, df = 5, p-value = 0.007422
```

Limitations of the Chi-Square Test

1. **Minimum Expected Frequency**: If $E_i < 5$ for any category, the test may lose power. Consider merging categories to meet this criterion.
2. **Independence**: Assumes observations are independent. Violations of

this assumption can invalidate the test.

3. **Sample Size Sensitivity**: Large sample sizes may result in significant $\chi 2$ values even for minor deviations from the expected distribution.

The Chi-Square Goodness-of-Fit Test is a versatile tool for evaluating the fit of observed data to a hypothesized distribution, widely used in fields like quality control, genetics, and market research.

4.6.4 Cramér-von Mises Test

The **Cramér-von Mises (CvM) Test** is a goodness-of-fit test that evaluates whether a sample data set comes from a specified distribution. Similar to the Kolmogorov-Smirnov Test (KS) and Anderson-Darling Test (AD), it assesses the discrepancy between the empirical and theoretical cumulative distribution functions (CDFs). However, the CvM test has **equal sensitivity across the entire distribution**, unlike the KS test (focused on the maximum difference) or the AD test (emphasizing the tails).

The Cramér-von Mises test statistic is defined as:

$$W^2 = n \int_{-\infty}^{\infty} \left(F_n(x) - F(x)\right)^2 dF(x)$$

Where:

- n is the sample size.
- $F_n(x)$ is the empirical cumulative distribution function (ECDF) of the sample.
- $F(x)$ is the CDF of the specified theoretical distribution.

For practical implementation, the test statistic is often computed as:

$$W^2 = \sum_{i=1}^{n} \left[F(X_i) - \frac{2i-1}{2n}\right]^2 + \frac{1}{12n}$$

Where X_i are the ordered sample values.

Hypotheses

- **Null Hypothesis (H_0)**: The sample data follow the specified distribution.
- **Alternative Hypothesis (H_a)**: The sample data do not follow the specified distribution.

Key Properties

1. **Equal Sensitivity**:
   - The CvM test gives equal weight to discrepancies across all parts of the distribution, unlike the AD test, which emphasizes the tails.
2. **Non-parametric**:
   - The test makes no strong parametric assumptions about the data, aside from the specified distribution.
3. **Complementary to KS and AD Tests**:
   - While the KS test focuses on the maximum distance between CDFs and the AD test emphasizes tails, the CvM test provides a balanced sensitivity across the entire range of the distribution.

The Cramér-von Mises test is widely used in:

1. **Goodness-of-Fit Testing**: Assessing whether data follow a specified theoretical distribution (e.g., normal, exponential).
2. **Model Validation**: Evaluating the fit of probabilistic models in statistical and machine learning contexts.
3. **Complementary Testing**: Used alongside KS and AD tests for a comprehensive analysis of distributional assumptions.

Example 1: Testing Normality

```r
library(nortest)

# Generate a sample from a normal distribution
set.seed(1)
sample_data <- rnorm(100, mean = 0, sd = 1)

# Perform the Cramér-von Mises test for normality
cvm_test_result <- cvm.test(sample_data)
print(cvm_test_result)
#>
#>  Cramer-von Mises normality test
#>
#> data:  sample_data
#> W = 0.026031, p-value = 0.8945
```

The test evaluates whether the sample data follow a normal distribution.

Example 2: Goodness-of-Fit for Custom Distributions

For distributions other than normal, you can use resampling techniques or custom implementations. Here's a pseudo-implementation for a custom theoretical

distribution:

```
# Custom ECDF and theoretical CDF comparison
set.seed(1)
sample_data <-
    rexp(100, rate = 1)  # Sample from exponential distribution
theoretical_cdf <-
    function(x) {
        pexp(x, rate = 1)
    }  # Exponential CDF

# Compute empirical CDF
empirical_cdf <- ecdf(sample_data)

# Compute CvM statistic
cvm_statistic <-
    sum((empirical_cdf(sample_data) - theoretical_cdf(sample_data)) ^ 2) / length(sample_data)
print(paste("Cramér-von Mises Statistic (Custom):", round(cvm_statistic, 4)))
#> [1] "Cramér-von Mises Statistic (Custom): 0.0019"
```

This demonstrates a custom calculation of the CvM statistic for testing goodness-of-fit to an exponential distribution.

1. **Normality Test**:
   - The `cvm.test` function evaluates whether the sample data follow a normal distribution.
   - A small p-value indicates significant deviation from normality.
2. **Custom Goodness-of-Fit**:
   - Custom implementation allows testing for distributions other than normal.
   - The statistic measures the squared differences between the empirical and theoretical CDFs.

Advantages and Limitations

1. **Advantages**:
   - Balanced sensitivity across the entire distribution.
   - Complements KS and AD tests by providing a different perspective on goodness-of-fit.
2. **Limitations**:
   - Critical values are distribution-specific.
   - The test may be less sensitive to tail deviations compared to the AD test.

The Cramér-von Mises test is a robust and versatile goodness-of-fit test, offering balanced sensitivity across the entire distribution. Its complementarity to KS and AD tests makes it an essential tool for validating distributional assumptions in both theoretical and applied contexts.

4.6.5 Kullback-Leibler Divergence

Kullback-Leibler (KL) divergence, also known as **relative entropy**, is a measure used to quantify the similarity between two probability distributions. It plays a critical role in statistical inference, machine learning, and information theory. However, KL divergence is not a true metric as it does not satisfy the triangle inequality.

Key Properties of KL Divergence

- **Not a Metric**: KL divergence fails to meet the triangle inequality requirement, and it is not symmetric, meaning:

$$D_{KL}(P\|Q) \neq D_{KL}(Q\|P)$$

- **Generalization to Multivariate Case**: KL divergence can be extended for multivariate distributions, making it flexible for complex analyses.
- **Quantifies Information Loss**: It measures the "information loss" when approximating the true distribution P with the predicted distribution Q. Thus, smaller values indicate closer similarity between the distributions.

Mathematical Definitions

KL divergence is defined differently for discrete and continuous distributions.

1. Discrete Case
For two discrete probability distributions $P = \{P_i\}$ and $Q = \{Q_i\}$, the KL divergence is given by:

$$D_{KL}(P\|Q) = \sum_i P_i \log\left(\frac{P_i}{Q_i}\right)$$

2. Continuous Case
For continuous probability density functions $P(x)$ and $Q(x)$:

$$D_{KL}(P\|Q) = \int P(x) \log\left(\frac{P(x)}{Q(x)}\right) dx$$

- **Range**:

$$D_{KL}(P\|Q) \in [0, \infty)$$

  - $D_{KL} = 0$ indicates identical distributions ($P = Q$).
  - Larger values indicate greater dissimilarity between P and Q.
- **Non-Symmetric Nature**: As noted, $D_{KL}(P\|Q)$ and $D_{KL}(Q\|P)$ are not equal, emphasizing its directed nature.

```r
library(philentropy)

# Example 1: Continuous case
# Define two continuous probability distributions with distinct patterns
X_continuous <- c(0.1, 0.2, 0.3, 0.4)  # Normalized to sum to 1
Y_continuous <- c(0.4, 0.3, 0.2, 0.1)  # Normalized to sum to 1

# Calculate KL divergence (logarithm base 2)
KL_continuous <- KL(rbind(X_continuous, Y_continuous), unit = "log2")
print(paste("KL divergence (continuous):", round(KL_continuous, 2)))
#> [1] "KL divergence (continuous): 0.66"

# Example 2: Discrete case
# Define two discrete probability distributions
X_discrete <- c(5, 10, 15, 20)  # Counts for events
Y_discrete <- c(20, 15, 10, 5)  # Counts for events

# Estimate probabilities empirically and compute KL divergence
KL_discrete <- KL(rbind(X_discrete, Y_discrete), est.prob = "empirical")
print(paste("KL divergence (discrete):", round(KL_discrete, 2)))
#> [1] "KL divergence (discrete): 0.66"
```

Insights:

1. Continuous case uses normalized probability values explicitly provided.
2. Discrete case relies on empirical estimation of probabilities from counts.
3. Observe how KL divergence quantifies the "distance" between the two distributions.

4.6.6 Jensen-Shannon Divergence

Jensen-Shannon (JS) divergence is a symmetric and bounded measure of the similarity between two probability distributions. It is derived from the Kullback-Leibler Divergence (KL) but addresses its asymmetry and unboundedness by incorporating a mixed distribution.

The Jensen-Shannon divergence is defined as:

$$D_{JS}(P\|Q) = \frac{1}{2}\left(D_{KL}(P\|M) + D_{KL}(Q\|M)\right)$$

where:

- $M = \frac{1}{2}(P + Q)$ is the **mixed distribution**, representing the average of P and Q.
- D_{KL} is the Kullback-Leibler divergence.

Key Properties

- **Symmetry**: Unlike KL divergence, JS divergence is symmetric:

$$D_{JS}(P\|Q) = D_{JS}(Q\|P)$$

- **Boundedness**:
  - For base-2 logarithms:
$$D_{JS} \in [0, 1]$$
  - For natural logarithms (base-e):
$$D_{JS} \in [0, \ln(2)]$$
- **Interpretability**: The JS divergence measures the average information gain when moving from the mixed distribution M to either P or Q. Its bounded nature makes it easier to compare across datasets.

```r
# Load the required library
library(philentropy)

# Example 1: Continuous case
# Define two continuous distributions
X_continuous <- 1:10  # Continuous sequence
Y_continuous <- 1:20  # Continuous sequence

# Compute JS divergence (logarithm base 2)
JS_continuous <- JSD(rbind(X_continuous, Y_continuous), unit = "log2")
print(paste("JS divergence (continuous):", round(JS_continuous, 2)))
#> [1] "JS divergence (continuous): 20.03"

# X_continuous and Y_continuous represent continuous distributions.
# The mixed distribution (M) is computed internally as the average of the two distribu

# Example 2: Discrete case
# Define two discrete distributions
X_discrete <- c(5, 10, 15, 20)  # Observed counts for events
Y_discrete <- c(20, 15, 10, 5)  # Observed counts for events

# Compute JS divergence with empirical probability estimation
JS_discrete <- JSD(rbind(X_discrete, Y_discrete), est.prob = "empirical")
print(paste("JS divergence (discrete):", round(JS_discrete, 2)))
```

```
#> [1] "JS divergence (discrete): 0.15"

# X_discrete and Y_discrete represent event counts.
# Probabilities are estimated empirically before calculating the divergence.
```

4.6.7 Hellinger Distance

The Hellinger distance is a bounded and symmetric measure of similarity between two probability distributions. It is widely used in statistics and machine learning to quantify how "close" two distributions are, with values ranging between 0 (identical distributions) and 1 (completely disjoint distributions).

Mathematical Definition

The Hellinger distance between two probability distributions P and Q is defined as:

$$H(P,Q) = \frac{1}{\sqrt{2}}\sqrt{\sum_x \left(\sqrt{P(x)} - \sqrt{Q(x)}\right)^2}$$

Where:

- $P(x)$ and $Q(x)$ are the probability densities or probabilities at point x for the distributions P and Q.
- The term $\sqrt{P(x)}$ is the square root of the probabilities, emphasizing geometric comparisons between the distributions.

Alternatively, for continuous distributions, the Hellinger distance can be expressed as:

$$H(P,Q) = \frac{1}{\sqrt{2}}\sqrt{\int \left(\sqrt{P(x)} - \sqrt{Q(x)}\right)^2 dx}$$

Key Properties

1. **Symmetry**:
$$H(P,Q) = H(Q,P)$$
The distance is symmetric, unlike Kullback-Leibler divergence.

2. **Boundedness**:
$$H(P,Q) \in [0,1]$$
   - $H = 0$: The distributions are identical ($P(x) = Q(x)$ for all x).
   - $H = 1$: The distributions have no overlap ($P(x) \neq Q(x)$).

3. **Interpretability**:

    - Hellinger distance provides a scale-invariant measure, making it suitable for comparing distributions in various contexts.

Hellinger distance is widely used in:

- **Hypothesis Testing**: Comparing empirical distributions to theoretical models.
- **Machine Learning**: Feature selection, classification, and clustering tasks.
- **Bayesian Analysis**: Quantifying differences between prior and posterior distributions.
- **Economics and Ecology**: Measuring dissimilarity in distributions like income, species abundance, or geographical data.

```r
library(philentropy)

# Example 1: Compute Hellinger Distance for Discrete Distributions
# Define two discrete distributions as probabilities
P_discrete <- c(0.1, 0.2, 0.3, 0.4)  # Normalized probabilities
Q_discrete <- c(0.3, 0.3, 0.2, 0.2)  # Normalized probabilities

# Compute Hellinger distance
hellinger_discrete <- distance(rbind(P_discrete, Q_discrete), method = "hellinger")
print(paste("Hellinger Distance (Discrete):", round(hellinger_discrete, 4)))
#> [1] "Hellinger Distance (Discrete): 0.465"

# Example 2: Compute Hellinger Distance for Empirical Distributions
# Define two empirical distributions (counts)
P_empirical <- c(10, 20, 30, 40)  # Counts for distribution P
Q_empirical <- c(30, 30, 20, 20)  # Counts for distribution Q

# Normalize counts to probabilities
P_normalized <- P_empirical / sum(P_empirical)
Q_normalized <- Q_empirical / sum(Q_empirical)

# Compute Hellinger distance
hellinger_empirical <- distance(rbind(P_normalized, Q_normalized), method = "hellinger"
print(paste("Hellinger Distance (Empirical):", round(hellinger_empirical, 4)))
#> [1] "Hellinger Distance (Empirical): 0.465"
```

4.6.8 Bhattacharyya Distance

The **Bhattacharyya Distance** is a statistical measure used to quantify the similarity or overlap between two probability distributions. It is commonly used in pattern recognition, signal processing, and statistics to evaluate how closely related two distributions are. The Bhattacharyya distance is particularly effective for comparing both discrete and continuous distributions.

The Bhattacharyya distance between two probability distributions P and Q is defined as:

$$D_B(P, Q) = -\ln\left(\sum_x \sqrt{P(x)Q(x)}\right)$$

For continuous distributions, the Bhattacharyya distance is expressed as:

$$D_B(P, Q) = -\ln\left(\int \sqrt{P(x)Q(x)}dx\right)$$

Where:

- $P(x)$ and $Q(x)$ are the probability densities or probabilities for the distributions P and Q.
- The term $\int \sqrt{P(x)Q(x)}dx$ is known as the **Bhattacharyya coefficient**.

Key Properties

1. **Symmetry**:
$$D_B(P, Q) = D_B(Q, P)$$
2. **Range**:
$$D_B(P, Q) \in [0, \infty)$$
   - $D_B = 0$: The distributions are identical ($P = Q$).
   - Larger values indicate less overlap and greater dissimilarity between P and Q.
3. **Relation to Hellinger Distance**:
   - The Bhattacharyya coefficient is related to the Hellinger distance:
$$H(P, Q) = \sqrt{1 - \sum_x \sqrt{P(x)Q(x)}}$$

The Bhattacharyya distance is widely used in:

1. **Classification**: Measuring the similarity between feature distributions in machine learning.
2. **Hypothesis Testing**: Evaluating the closeness of observed data to a theoretical model.
3. **Image Processing**: Comparing pixel intensity distributions or color histograms.
4. **Economics and Ecology**: Assessing similarity in income distributions or species abundance.

Example 1: Discrete Distributions

```
# Define two discrete probability distributions
P_discrete <- c(0.1, 0.2, 0.3, 0.4)  # Normalized probabilities
Q_discrete <- c(0.3, 0.3, 0.2, 0.2)  # Normalized probabilities

# Compute Bhattacharyya coefficient
bhattacharyya_coefficient <- sum(sqrt(P_discrete * Q_discrete))

# Compute Bhattacharyya distance
bhattacharyya_distance <- -log(bhattacharyya_coefficient)

# Display results
print(paste(
    "Bhattacharyya Coefficient:",
    round(bhattacharyya_coefficient, 4)
))
#> [1] "Bhattacharyya Coefficient: 0.9459"
print(paste(
    "Bhattacharyya Distance (Discrete):",
    round(bhattacharyya_distance, 4)
))
#> [1] "Bhattacharyya Distance (Discrete): 0.0556"
```

A smaller Bhattacharyya distance indicates greater similarity between the two distributions.

Example 2: Continuous Distributions (Approximation)

For continuous distributions, the Bhattacharyya distance can be approximated using numerical integration or discretization.

```
# Generate two continuous distributions
set.seed(1)
P_continuous <-
    rnorm(1000, mean = 0, sd = 1)  # Standard normal distribution
Q_continuous <-
    rnorm(1000, mean = 1, sd = 1)  # Normal distribution with mean 1
```

```r
# Create histograms to approximate probabilities
hist_P <- hist(P_continuous, breaks = 50, plot = FALSE)
hist_Q <- hist(Q_continuous, breaks = 50, plot = FALSE)

# Normalize histograms to probabilities
prob_P <- hist_P$counts / sum(hist_P$counts)
prob_Q <- hist_Q$counts / sum(hist_Q$counts)

# Compute Bhattacharyya coefficient
bhattacharyya_coefficient_continuous <- sum(sqrt(prob_P * prob_Q))

# Compute Bhattacharyya distance
bhattacharyya_distance_continuous <-
    -log(bhattacharyya_coefficient_continuous)

# Display results
print(paste(
    "Bhattacharyya Coefficient (Continuous):",
    round(bhattacharyya_coefficient_continuous, 4)
))
#> [1] "Bhattacharyya Coefficient (Continuous): 0.9823"
print(paste(
    "Bhattacharyya Distance (Continuous Approximation):",
    round(bhattacharyya_distance_continuous, 4)
))
#> [1] "Bhattacharyya Distance (Continuous Approximation): 0.0178"
```

Continuous distributions are discretized into histograms to compute the Bhattacharyya coefficient and distance.

1. Discrete Case:
    1. The Bhattacharyya coefficient quantifies the overlap between P and Q.
    2. The Bhattacharyya distance translates this overlap into a logarithmic measure of dissimilarity.
2. Continuous Case:
    1. Distributions are discretized into histograms to approximate the Bhattacharyya coefficient and distance.

4.6.9 Wasserstein Distance

The Wasserstein distance, also known as the **Earth Mover's Distance (EMD)**, is a measure of similarity between two probability distributions. It quantifies the "cost" of transforming one distribution into another, making it particularly suitable for continuous data and applications where the geometry of the data matters.

Mathematical Definition

The Wasserstein distance between two probability distributions P and Q over a domain $\mathcal{X}$ is defined as:

$$W_p(P, Q) = \left(\int_{\mathcal{X}} |F_P(x) - F_Q(x)|^p dx \right)^{\frac{1}{p}}$$

Where:

- $F_P(x)$ and $F_Q(x)$ are the cumulative distribution functions (CDFs) of P and Q.
- $p \geq 1$ is the order of the Wasserstein distance (commonly $p = 1$).
- $|\cdot|^p$ is the absolute difference raised to the power p.

For the case of $p = 1$, the formula simplifies to:

$$W_1(P, Q) = \int_{\mathcal{X}} |F_P(x) - F_Q(x)| dx$$

This represents the minimum "cost" of transforming the distribution P into Q, where cost is proportional to the distance a "unit of mass" must move.

Key Properties

- **Interpretability**: Represents the "effort" required to morph one distribution into another.
- **Metric**: Wasserstein distance satisfies the properties of a metric, including symmetry, non-negativity, and the triangle inequality.
- **Flexibility**: Can handle both empirical and continuous distributions.

Wasserstein distance is widely used in various fields, including:

1. **Machine Learning**:
   - Training generative models such as Wasserstein GANs.
   - Monitoring data drift in online systems.
2. **Statistics**:
   - Comparing empirical distributions derived from observed data.
   - Robustness testing under distributional shifts.
3. **Economics**:
   - Quantifying disparities in income or wealth distributions.
4. **Image Processing**:
   - Measuring structural differences between image distributions.

```r
library(transport)
library(twosamples)

# Example 1: Compute Wasserstein Distance (1D case)
set.seed(1)
dist_1 <- rnorm(100)               # Generate a sample from a standard normal distribution
dist_2 <- rnorm(100, mean = 1)     # Generate a sample with mean shifted to 1

# Calculate the Wasserstein distance
wass_distance <- wasserstein1d(dist_1, dist_2)
print(paste("1D Wasserstein Distance:", round(wass_distance, 4)))
#> [1] "1D Wasserstein Distance: 0.8533"

# Example 2: Wasserstein Metric as a Statistic
set.seed(1)
wass_stat_value <- wass_stat(dist_1, dist_2)
print(paste("Wasserstein Statistic:", round(wass_stat_value, 4)))
#> [1] "Wasserstein Statistic: 0.8533"

# Example 3: Wasserstein Test (Permutation-based Two-sample Test)
set.seed(1)
wass_test_result <- wass_test(dist_1, dist_2)
print(wass_test_result)
#> Test Stat   P-Value
#> 0.8533046 0.0002500

# - Example 1 calculates the simple Wasserstein distance between two distributions.
# - Example 2 computes the Wasserstein distance as a statistical metric.
# - Example 3 performs a permutation-based two-sample test using the Wasserstein metric.
```

4.6.10 Energy Distance

The **Energy Distance** is a statistical metric used to quantify the similarity between two probability distributions. It is particularly effective for comparing multi-dimensional distributions.

The Energy Distance between two distributions P and Q is defined as:

$$E(P, Q) = 2\mathbb{E}[||X - Y||] - \mathbb{E}[||X - X'||] - \mathbb{E}[||Y - Y'||]$$

Where:

- X and X' are independent and identically distributed (i.i.d.) random variables from P.
- Y and Y' are i.i.d. random variables from Q.
- $||\cdot||$ denotes the Euclidean distance.

Alternatively, for empirical distributions, the Energy Distance can be approximated as:

$$E(P,Q) = \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}||X_i - Y_j|| - \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}||X_i - X_j|| - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}||Y_i - Y_j||$$

Where:

- m and n are the sample sizes from distributions P and Q respectively.
- X_i and Y_j are samples from P and Q.

Key Properties

1. **Metric**:
   - Energy distance satisfies the properties of a metric: symmetry, non-negativity, and the triangle inequality.
2. **Range**:
   $$E(P,Q) \geq 0$$
   - $E(P,Q) = 0$: The distributions are identical.
   - Larger values indicate greater dissimilarity.
3. **Effectiveness for Multi-dimensional Data**:
   - Energy distance is designed to work well in higher-dimensional spaces, unlike some traditional metrics.

The Energy Distance is widely used in:

1. **Hypothesis Testing**: Testing whether two distributions are the same.
2. **Energy Test** for equality of distributions.
3. **Clustering**: Measuring dissimilarity between clusters in multi-dimensional data.
4. **Feature Selection**: Comparing distributions of features across different classes to evaluate their discriminative power.

Example 1: Comparing Two Distributions

```r
# Load the 'energy' package
library(energy)

# Generate two sample distributions
set.seed(1)
X <- matrix(rnorm(1000, mean = 0, sd = 1), ncol = 2)  # Distribution P
Y <- matrix(rnorm(1000, mean = 1, sd = 1), ncol = 2)  # Distribution Q

# Combine X and Y and create a group identifier
combined <- rbind(X, Y)
groups <- c(rep(1, nrow(X)), rep(2, nrow(Y)))

# Compute Energy Distance
energy_dist <- edist(combined, sizes = table(groups))

# Print the Energy Distance
print(paste("Energy Distance:", round(energy_dist, 4)))
#> [1] "Energy Distance: 201.9202"
```

This calculates the energy distance between two multi-dimensional distributions.

Example 2: Energy Test for Equality of Distributions

```r
# Perform the Energy Test
energy_test <-
    eqdist.etest(rbind(X, Y), sizes = c(nrow(X), nrow(Y)), R = 999)
print(energy_test)
#>
#>  Multivariate 2-sample E-test of equal distributions
#>
#> data:  sample sizes 500 500, replicates 999
#> E-statistic = 201.92, p-value = 0.001
```

The energy test evaluates the null hypothesis that the two distributions are identical.

1. **Energy Distance**:
   - Provides a single metric to quantify the dissimilarity between two distributions, considering all dimensions of the data.
2. **Energy Test**:
   - Tests for equality of distributions using Energy Distance.
   - The p-value indicates whether the distributions are significantly different.

Advantages of Energy Distance

1. **Multi-dimensional Applicability**:
   - Works seamlessly with high-dimensional data, unlike some divergence metrics which may suffer from dimensionality issues.
2. **Non-parametric**:
   - Makes no assumptions about the form of the distributions.
3. **Robustness**:
   - Effective even with complex data structures.

4.6.11 Total Variation Distance

The **Total Variation (TV) Distance** is a measure of the maximum difference between two probability distributions. It is widely used in probability theory, statistics, and machine learning to quantify how dissimilar two distributions are.

The Total Variation Distance between two probability distributions P and Q is defined as:

$$D_{TV}(P, Q) = \frac{1}{2} \sum_x |P(x) - Q(x)|$$

Where:

- $P(x)$ and $Q(x)$ are the probabilities assigned to the outcome x by the distributions P and Q.
- The factor $\frac{1}{2}$ ensures that the distance lies within the range $[0, 1]$.

Alternatively, for continuous distributions, the TV distance can be expressed as:

$$D_{TV}(P, Q) = \frac{1}{2} \int |P(x) - Q(x)| dx$$

Key Properties

1. **Range**:
$$D_{TV}(P, Q) \in [0, 1]$$
   - $D_{TV} = 0$: The distributions are identical ($P = Q$).
   - $D_{TV} = 1$: The distributions are completely disjoint (no overlap).
2. **Symmetry**:
$$D_{TV}(P, Q) = D_{TV}(Q, P)$$
3. **Interpretability**:

- $D_{TV}(P, Q)$ represents the maximum probability mass that needs to be shifted to transform P into Q.

The Total Variation Distance is used in:

1. **Hypothesis Testing**: Quantifying the difference between observed and expected distributions.
2. **Machine Learning**: Evaluating similarity between predicted and true distributions.
3. **Information Theory**: Comparing distributions in contexts like communication and cryptography.

Example 1: Discrete Distributions

```
# Define two discrete probability distributions
P_discrete <- c(0.1, 0.2, 0.3, 0.4)  # Normalized probabilities
Q_discrete <- c(0.3, 0.3, 0.2, 0.2)  # Normalized probabilities

# Compute Total Variation Distance
tv_distance <- sum(abs(P_discrete - Q_discrete)) / 2
print(paste("Total Variation Distance (Discrete):", round(tv_distance, 4)))
#> [1] "Total Variation Distance (Discrete): 0.3"
```

This calculates the maximum difference between the two distributions, scaled to lie between 0 and 1.

Example 2: Continuous Distributions (Approximation)

For continuous distributions, the TV distance can be approximated using discretization or numerical integration. Here's an example using random samples:

```
# Generate two continuous distributions
set.seed(1)
P_continuous <-
    rnorm(1000, mean = 0, sd = 1)  # Standard normal distribution
Q_continuous <-
    rnorm(1000, mean = 1, sd = 1)  # Normal distribution with mean 1

# Create histograms to approximate probabilities
hist_P <- hist(P_continuous, breaks = 50, plot = FALSE)
hist_Q <- hist(Q_continuous, breaks = 50, plot = FALSE)

# Normalize histograms to probabilities
prob_P <- hist_P$counts / sum(hist_P$counts)
prob_Q <- hist_Q$counts / sum(hist_Q$counts)
```

```r
# Compute Total Variation Distance
tv_distance_continuous <- sum(abs(prob_P - prob_Q)) / 2
print(paste(
    "Total Variation Distance (Continuous Approximation):",
    round(tv_distance_continuous, 4)
))
#> [1] "Total Variation Distance (Continuous Approximation): 0.125"
```

The continuous distributions are discretized into histograms, and TV distance is computed based on the resulting probabilities.

1. **Discrete Case**:
   - The TV distance quantifies the maximum difference between P and Q in terms of probability mass.
   - In this example, it highlights how much P and Q diverge.
2. **Continuous Case**:
   - For continuous distributions, TV distance is approximated using discretized probabilities from histograms.
   - This approach provides an intuitive measure of similarity for large samples.

The Total Variation Distance provides an intuitive and interpretable measure of the maximum difference between two distributions. Its symmetry and bounded nature make it a versatile tool for comparing both discrete and continuous distributions.

4.6.12 Summary

1. Tests for Comparing Distributions

Test Name	Purpose	Type of Data	Advantages	Limitations
Kolmogorov-Smirnov Test	Tests if two distributions are the same or if a sample matches a reference distribution.	Empirical Distributions (Continuous)	Non-parametric, detects global differences.	Less sensitive to tail differences, limited to one-dimensional data.
Anderson-Darling Test	Tests goodness-of-fit with emphasis on the tails.	Continuous Data	Strong sensitivity to tail behavior.	Requires specifying a reference distribution.

Test Name	Purpose	Type of Data	Advantages	Limitations
Chi-Square Goodness-of-Fit Test	Tests if observed frequencies match expected frequencies.	Categorical Data	Simple, intuitive for discrete data.	Requires large sample sizes and sufficiently large expected frequencies.
Cramér-von Mises Test	Evaluates goodness-of-fit using cumulative distribution functions.	Empirical Distributions (Continuous)	Sensitive across the entire distribution.	Limited to one-dimensional data; requires cumulative distribution functions.

2. Divergence Metrics

Metric Name	Purpose	Type of Data	Advantages	Limitations
Kullback-Leibler Divergence	Measures how one probability distribution diverges from another.	Probability Distributions (Continuous/Discrete)	Provides a clear measure of information loss.	Asymmetric, sensitive to zero probabilities.
Jensen-Shannon Divergence	Symmetric measure of similarity between two probability distributions.	Probability Distributions (Continuous/Discrete)	Symmetric and bounded; intuitive for comparison.	Less sensitive to tail differences.
Hellinger Distance	Measures geometric similarity between two probability distributions.	Discrete or Continuous Probability Distributions	Easy to interpret; bounded between 0 and 1.	Computationally expensive for large datasets.
Bhattacharyya Distance	Quantifies overlap between two statistical distributions.	Probability Distributions (Continuous/Discrete)	Useful for classification and clustering tasks.	Less interpretable in large-scale applications.

3. Distance Metrics

Metric Name	Purpose	Type of Data	Advantages	Limitations
Wasserstein Distance	Measures the "effort" or "cost" to transform one distribution into another.	Continuous or Empirical Distributions	Provides geometric interpretation; versatile.	Computationally expensive for large-scale data.
Energy Distance	Measures statistical dissimilarity between multivariate distributions.	Multivariate Empirical Distributions	Non-parametric, works well for high-dimensional data.	Requires pairwise calculations; sensitive to outliers.
Total Variation Distance	Measures the maximum absolute difference between probabilities of two distributions.	Probability Distributions (Discrete/Continuous)	Intuitive and strict divergence measure.	Ignores structural differences beyond the largest deviation.

II. REGRESSION

Chapter 5

Linear Regression

Linear regression is one of the most fundamental tools in statistics and econometrics, widely used for modeling relationships between variables. It forms the cornerstone of predictive analysis, enabling us to understand and quantify how changes in one or more explanatory variables are associated with a dependent variable. Its simplicity and versatility make it an essential tool in fields ranging from economics and marketing to healthcare and environmental studies.

At its core, linear regression addresses questions about associations rather than causation. For example:

- How are advertising expenditures associated with sales performance?
- What is the relationship between a company's revenue and its stock price?
- How does the level of education correlate with income?

These questions are about patterns in data—not necessarily causal effects. While regression can provide insights into potential causal relationships, establishing causality requires more than just regression analysis. It requires careful consideration of the study design, assumptions, and potential confounding factors.

So, why is it called "linear"? The term refers to the structure of the model, where the dependent variable (outcome) is modeled as a linear combination of one or more independent variables (predictors). For example, in simple linear regression, the relationship is represented as:

$$Y = \beta_0 + \beta_1 X + \epsilon,$$

where Y is the dependent variable, X is the independent variable, β_0 and β_1 are parameters to be estimated, and ϵ is the error term capturing randomness or unobserved factors.

Linear regression serves as a foundation for much of applied data analysis because of its wide-ranging applications:

1. **Understanding Patterns in Data:** Regression provides a framework to summarize and explore relationships between variables. It allows us to identify patterns such as trends or associations, which can guide further analysis or decision-making.
2. **Prediction:** Beyond exploring relationships, regression is widely used for making predictions. For instance, given historical data, we can use a regression model to predict future outcomes like sales, prices, or demand.
3. **Building Blocks for Advanced Techniques:** Linear regression is foundational for many advanced statistical and machine learning models, such as logistic regression, ridge regression, and neural networks. Mastering linear regression equips you with the skills to tackle more complex methods.

Regression and Causality: A Crucial Distinction

It's essential to remember that regression alone does not establish causation. For instance, a regression model might show a strong association between advertising and sales, but this does not prove that advertising directly causes sales to increase. Other factors—such as seasonality, market trends, or unobserved variables—could also influence the results.

Establishing causality requires additional steps, such as controlled experiments, instrumental variable techniques, or careful observational study designs. As we work through the details of linear regression, we'll revisit this distinction and highlight scenarios where causality might or might not be inferred.

What is an Estimator?

At the heart of regression lies the process of **estimation**—the act of using data to determine the unknown characteristics of a population or model.

An **estimator** is a mathematical rule or formula used to calculate an estimate of an unknown quantity based on observed data. For example, when we calculate

the average height of a sample to estimate the average height of a population, the sample mean is the estimator.

In the context of regression, the quantities we typically estimate are:

- **Parameters**: Fixed, unknown values that describe the relationship between variables (e.g., coefficients in a regression equation).
  - **Estimating parameters** → Parametric models (finite parameters, e.g., coefficients in regression).
- **Functions**: Unknown relationships or patterns in the data, often modeled without assuming a fixed functional form.
  - **Estimating functions** → Non-parametric models (focus on shapes or trends, not a fixed number of parameters).

Types of Estimators

To better understand the estimation process, let's introduce two broad categories of estimators that we'll work with:

1. **Parametric Estimators**
   Parametric estimation focuses on a finite set of parameters that define a model. For example, in a simple linear regression:

   $$Y = \beta_0 + \beta_1 X + \epsilon,$$

   the task is to estimate the parameters β_0 (intercept) and β_1 (slope). Parametric estimators rely on specific assumptions about the form of the model (e.g., linearity) and the distribution of the error term (e.g., normality).

2. **Non-Parametric Estimators**
   Non-parametric estimation avoids assuming a specific functional form for the relationship between variables. Instead, it focuses on estimating patterns or trends directly from the data. For example, using a scatterplot smoothing technique to visualize how sales vary with advertising spend without imposing a linear or quadratic relationship.

These two categories reflect a fundamental trade-off in statistical analysis: **parametric models** are often simpler and more interpretable but require strong assumptions, while **non-parametric models** are more flexible but may require more data and computational resources.

Desirable Properties of Estimators

Regardless of whether we are estimating parameters or functions, we want our estimators to possess certain desirable properties. Think of these as the "golden standards" that help us judge whether an estimator is reliable:

1. **Unbiasedness**
   An estimator is unbiased if it hits the true value of the parameter, on average, over repeated samples. Mathematically:

   $$E[\hat{\beta}] = \beta.$$

   This means that, across multiple samples, the estimator does not systematically overestimate or underestimate the true parameter.

2. **Consistency**
   Consistency ensures that as the sample size increases, the estimator converges to the true value of the parameter. Formally:

   $$plim\ \hat{\beta}_n = \beta.$$

   This property relies on the Law of Large Numbers, which guarantees that larger samples reduce random fluctuations, leading to more precise estimates.

3. **Efficiency**
   Among all unbiased estimators, an efficient estimator has the smallest variance.

   - The Ordinary Least Squares method is efficient because it is the **Best Linear Unbiased Estimator (BLUE)** under the Gauss-Markov Theorem.
   - For estimators that meet specific distributional assumptions (e.g., normality), **Maximum Likelihood Estimators (MLE)** are asymptotically efficient, meaning they achieve the lowest possible variance as the sample size grows.

Why These Properties Matter

Understanding these properties is crucial because they ensure that the methods we use for estimation are reliable, precise, and robust. Whether we are estimating coefficients in a regression model or uncovering a complex pattern in the data, these properties provide the foundation for statistical inference and decision-making.

Now that we've established what estimators are, the types we'll encounter, and their desirable properties, we can move on to understanding how these concepts apply specifically to the Ordinary Least Squares method—the backbone of linear regression.

Reference Table

Estimator	Key Assumptions	Strengths	Limitations
Ordinary Least Squares	Errors are independent, identically distributed (i.i.d.) with mean 0 and constant variance. Linear relationship between predictors and response.	Simple, well-understood method. Minimizes residual sum of squares (easy to interpret coefficients).	Sensitive to outliers and violations of normality. Can perform poorly if predictors are highly correlated (multicollinearity).
Generalized Least Squares	Errors have a known correlation structure or heteroscedasticity structure that can be modeled.	Handles correlated or non-constant-variance errors. More flexible than OLS when noise structure is known.	Requires specifying (or estimating) the error covariance structure. Misspecification can lead to biased estimates.
Maximum Likelihood	Underlying probability distribution (e.g., normal) must be specified correctly.	Provides a general framework for estimating parameters under well-defined probability models. Can extend to complex likelihoods.	Highly sensitive to model misspecification. May require more computation than OLS or GLS.
Penalized (Regularized) Estimators	Coefficients assumed to be shrinkable; model typically allows coefficient penalization.	Controls overfitting via regularization. Handles high-dimensional data or many predictors. Can perform feature selection (e.g., Lasso).	Requires choosing tuning parameter(s) (e.g.,). Interpretation of coefficients becomes less straightforward.
Robust Estimators	Less sensitive to heavy-tailed or outlier-prone distributions (weaker assumptions on the error structure).	Resistant to large deviations or outliers in data. Often maintains good performance under mild model misspecifications.	Less efficient if errors are truly normal. Choice of robust method and tuning can be subjective.

Estimator	Key Assumptions	Strengths	Limitations
Partial Least Squares	Predictors may be highly correlated; dimension reduction is desired.	Simultaneously reduces dimensionality and fits regression. Works well with collinear, high-dimensional data.	Can be harder to interpret than OLS (latent components instead of original predictors). Requires choosing the number of components.

5.1 Ordinary Least Squares

Ordinary Least Squares (OLS) is the backbone of statistical modeling, a method so foundational that it often serves as the starting point for understanding data relationships. Whether predicting sales, estimating economic trends, or uncovering patterns in scientific research, OLS remains a critical tool. Its appeal lies in simplicity: OLS models the relationship between a dependent variable and one or more predictors by minimizing the squared differences between observed and predicted values.

Why OLS Works: Linear and Nonlinear Relationships

OLS rests on the Conditional Expectation Function (CEF), $E[Y|X]$, which describes the expected value of Y given X. Regression shines in two key scenarios:

1. **Perfect Fit (Linear CEF):**
   If $E[Y_i|X_{1i}, \dots, X_{Ki}] = a + \sum_{k=1}^{K} b_k X_{ki}$, the regression of Y_i on $X_{1i}, \dots, X_{Ki}$ exactly equals the CEF. In other words, the regression gives the true average relationship between Y and X.
   If the true relationship is linear, regression delivers the exact CEF. For instance, imagine you're estimating the relationship between advertising spend and sales revenue. If the true impact is linear, OLS will perfectly capture it.

2. **Approximation (Nonlinear CEF):**
   If $E[Y_i|X_{1i}, \dots, X_{Ki}]$ is nonlinear, OLS provides the best linear approximation to this relationship. Specifically, it minimizes the expected squared deviation between the linear regression line and the nonlinear CEF.
   For example, the effect of advertising diminishes at higher spending levels? OLS still works, providing the best linear approximation to this nonlinear relationship by minimizing the squared deviations between predictions and the true (but unknown) CEF.

In other words, regression is not just a tool for "linear" relationships—it's a workhorse that adapts remarkably well to messy, real-world data.

5.1.1 Simple Regression (Basic) Model

The simplest form of regression is a straight line:

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

where

- Y_i: The dependent variable or outcome we're trying to predict (e.g., sales, temperature).
- X_i: The independent variable or predictor (e.g., advertising spend, time).
- β_0: The intercept—where the line crosses the Y-axis when $X = 0$.
- β_1: The slope, representing the change in Y for a one-unit increase in X.
- ϵ_i: The error term, accounting for random factors that X cannot explain.

Assumptions About the Error Term (ϵ_i):

$$\begin{aligned} E(\epsilon_i) &= 0 \\ \text{Var}(\epsilon_i) &= \sigma^2 \\ \text{Cov}(\epsilon_i, \epsilon_j) &= 0 \quad \text{for all } i \neq j \end{aligned}$$

Since ϵ_i is random, Y_i is also random:

$$\begin{aligned} E(Y_i) &= E(\beta_0 + \beta_1 X_i + \epsilon_i) \\ &= \beta_0 + \beta_1 X_i \end{aligned}$$

$$\begin{aligned} \text{Var}(Y_i) &= \text{Var}(\beta_0 + \beta_1 X_i + \epsilon_i) \\ &= \text{Var}(\epsilon_i) \\ &= \sigma^2 \end{aligned}$$

Since $\text{Cov}(\epsilon_i, \epsilon_j) = 0$, the outcomes across observations are independent. Hence, Y_i and Y_j are uncorrelated as well, conditioned on the X's.

5.1.1.1 Estimation in Ordinary Least Squares

The goal of OLS is to estimate the regression parameters (β_0, β_1) that best describe the relationship between the dependent variable Y and the independent variable X. To achieve this, we minimize the sum of squared deviations between observed values of Y_i and their expected values predicted by the model.

The deviation of an observed value Y_i from its expected value, based on the regression model, is:

$$Y_i - E(Y_i) = Y_i - (\beta_0 + \beta_1 X_i).$$

This deviation represents the error in prediction for the i-th observation.

To ensure that the errors don't cancel each other out and to prioritize larger deviations, we consider the squared deviations. The sum of squared deviations, denoted by Q, is defined as:

$$Q = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_i)^2.$$

The goal of OLS is to find the values of β_0 and β_1 that minimize Q. These values are called the **OLS estimators**.

To minimize Q, we take partial derivatives with respect to β_0 and β_1, set them to zero, and solve the resulting system of equations. After simplifying, the estimators for the slope (b_1) and intercept (b_0) are obtained as follows:

Slope (b_1):

$$b_1 = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n} (X_i - \bar{X})^2}.$$

Here, $\bar{X}$ and $\bar{Y}$ represent the means of X and Y, respectively. This formula reveals that the slope is proportional to the covariance between X and Y, scaled by the variance of X.

Intercept (b_0):

$$b_0 = \frac{1}{n}\left(\sum_{i=1}^{n} Y_i - b_1 \sum_{i=1}^{n} X_i\right) = \bar{Y} - b_1\bar{X}.$$

The intercept is determined by aligning the regression line with the center of the data.

Intuition Behind the Estimators

- b_1 (Slope): This measures the average change in Y for a one-unit increase in X. The formula uses deviations from the mean to ensure that the relationship captures the joint variability of X and Y.
- b_0 (Intercept): This ensures that the regression line passes through the mean of the data points $(\bar{X}, \bar{Y})$, anchoring the model in the center of the observed data.

Equivalently, we can also write these parameters in terms of covariances.

The covariance between two variables is defined as:

$$\text{Cov}(X_i, Y_i) = E[(X_i - E[X_i])(Y_i - E[Y_i])]$$

Properties of Covariance:

1. $\text{Cov}(X_i, X_i) = \sigma_X^2$
2. If $E(X_i) = 0$ or $E(Y_i) = 0$, then $\text{Cov}(X_i, Y_i) = E[X_i Y_i]$
3. For $W_i = a + bX_i$ and $Z_i = c + dY_i$,
   $\text{Cov}(W_i, Z_i) = bd \cdot \text{Cov}(X_i, Y_i)$

For a bivariate regression, the slope β in a bivariate regression is given by:

$$\beta = \frac{\text{Cov}(Y_i, X_i)}{\text{Var}(X_i)}$$

For a multivariate case, the slope for X_k is:

$$\beta_k = \frac{\text{Cov}(Y_i, \tilde{X}_{ki})}{\text{Var}(\tilde{X}_{ki})}$$

Where $\tilde{X}_{ki}$ represents the residual from a regression of X_{ki} on the $K-1$ other covariates in the model.

The intercept is:

$$\beta_0 = E[Y_i] - \beta_1 E(X_i)$$

Note:

- OLS does not require the assumption of a specific distribution for the variables. Its robustness is based on the minimization of squared errors (i.e., no distributional assumptions).

5.1.1.2 Properties of Least Squares Estimators

The properties of the Ordinary Least Squares estimators (b_0 and b_1) are derived based on their statistical behavior. These properties provide insights into the accuracy, variability, and reliability of the estimates.

5.1.1.2.1 Expectation of the OLS Estimators The OLS estimators b_0 (intercept) and b_1 (slope) are unbiased. This means their expected values equal the true population parameters:

$$\begin{aligned} E(b_1) &= \beta_1, \\ E(b_0) &= E(\bar{Y}) - \bar{X}\beta_1. \end{aligned}$$

Since the expected value of the sample mean of Y, $E(\bar{Y})$, is:

$$E(\bar{Y}) = \beta_0 + \beta_1 \bar{X},$$

the expected value of b_0 simplifies to:

$$E(b_0) = \beta_0.$$

Thus, b_0 and b_1 are unbiased estimators of their respective population parameters β_0 and β_1.

5.1.1.2.2 Variance of the OLS Estimators The variability of the OLS estimators depends on the spread of the predictor variable X and the error variance σ^2. The variances are given by:

Variance of b_1 (Slope):

$$\text{Var}(b_1) = \frac{\sigma^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}.$$

Variance of b_0 (Intercept):

$$\text{Var}(b_0) = \sigma^2 \left(\frac{1}{n} + \frac{\bar{X}^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2} \right).$$

These formulas highlight that:

- $\text{Var}(b_1) \to 0$ as the number of observations increases, provided X_i values are distributed around their mean $\bar{X}$.
- $\text{Var}(b_0) \to 0$ as n increases, assuming X_i values are appropriately selected (i.e., not all clustered near the mean).

5.1.1.3 Mean Square Error (MSE)

The Mean Square Error (MSE) quantifies the average squared residual (error) in the model:

$$MSE = \frac{SSE}{n-2} = \frac{\sum_{i=1}^{n} e_i^2}{n-2} = \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n-2},$$

where SSE is the Sum of Squared Errors and $n-2$ represents the degrees of freedom for a simple linear regression model (two parameters estimated: β_0 and β_1).

The expected value of the MSE equals the error variance (i.e., unbiased Estimator of MSE:):

$$E(MSE) = \sigma^2.$$

5.1.1.4 Estimating Variance of the OLS Coefficients

The sample-based estimates of the variances of b_0 and b_1 are expressed as follows:

Estimated Variance of b_1 (Slope):

$$s^2(b_1) = \widehat{\text{Var}}(b_1) = \frac{MSE}{\sum_{i=1}^{n}(X_i - \bar{X})^2}.$$

Estimated Variance of b_0 (Intercept):

$$s^2(b_0) = \widehat{\text{Var}}(b_0) = MSE\left(\frac{1}{n} + \frac{\bar{X}^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}\right).$$

These estimates rely on the MSE to approximate σ^2.

The variance estimates are unbiased:

$$E(s^2(b_1)) = \text{Var}(b_1),$$
$$E(s^2(b_0)) = \text{Var}(b_0).$$

Implications of These Properties

1. **Unbiasedness:** The unbiased nature of b_0 and b_1 ensures that, on average, the regression model accurately reflects the true relationship in the population.
2. **Decreasing Variance:** As the sample size n increases or as the spread of X_i values grows, the variances of b_0 and b_1 decrease, leading to more precise estimates.
3. **Error Estimation with MSE:** MSE provides a reliable estimate of the error variance σ^2, which feeds directly into assessing the reliability of b_0 and b_1.

5.1.1.5 Residuals in Ordinary Least Squares

Residuals are the differences between observed values (Y_i) and their predicted counterparts ($\hat{Y}_i$). They play a central role in assessing model fit and ensuring the assumptions of OLS are met.

The residual for the i-th observation is defined as:

$$e_i = Y_i - \hat{Y}_i = Y_i - (b_0 + b_1 X_i),$$

where:

- e_i: Residual for the i-th observation.

- $\hat{Y}_i$: Predicted value based on the regression model.
- Y_i: Actual observed value.

Residuals estimate the unobservable error terms ϵ_i:

- e_i is an estimate of $\epsilon_i = Y_i - E(Y_i)$.
- ϵ_i is always unknown because we do not know the true values of β_0 and β_1.

5.1.1.5.1 Key Properties of Residuals Residuals exhibit several mathematical properties that align with the OLS estimation process:

1. **Sum of Residuals**:
   The residuals sum to zero:

$$\sum_{i=1}^{n} e_i = 0.$$

   This ensures that the regression line passes through the centroid of the data, $(\bar{X}, \bar{Y})$.

2. **Orthogonality of Residuals to Predictors**:
   The residuals are orthogonal (uncorrelated) to the predictor variable X:

$$\sum_{i=1}^{n} X_i e_i = 0.$$

   This reflects the fact that the OLS minimizes the squared deviations of residuals along the Y-axis, not the X-axis.

5.1.1.5.2 Expected Values of Residuals The expected values of residuals reinforce the unbiased nature of OLS:

1. **Mean of Residuals**:
   The residuals have an expected value of zero:

$$E[e_i] = 0.$$

2. **Orthogonality to Predictors and Fitted Values**:
   Residuals are uncorrelated with both the predictor variables and the fitted values:

$$E[X_i e_i] = 0,$$
$$E[\hat{Y}_i e_i] = 0.$$

These properties highlight that residuals do not contain systematic information about the predictors or the fitted values, reinforcing the idea that the model has captured the underlying relationship effectively.

5.1.1.5.3 Practical Importance of Residuals

1. **Model Diagnostics:**
   Residuals are analyzed to check the assumptions of OLS, including linearity, homoscedasticity (constant variance), and independence of errors. Patterns in residual plots can signal issues such as nonlinearity or heteroscedasticity.

2. **Goodness-of-Fit:**
   The sum of squared residuals, $\sum e_i^2$, measures the total unexplained variation in Y. A smaller sum indicates a better fit.

3. **Influence Analysis:**
   Large residuals may indicate outliers or influential points that disproportionately affect the regression line.

5.1.1.6 Inference in Ordinary Least Squares

Inference allows us to make probabilistic statements about the regression parameters (β_0, β_1) and predictions (Y_h). To perform valid inference, certain assumptions about the distribution of errors are necessary.

Normality Assumption

- OLS estimation itself does **not** require the assumption of normality.
- However, to conduct hypothesis tests or construct confidence intervals for β_0, β_1, and predictions, distributional assumptions are necessary.
- Inference on β_0 and β_1 is **robust** to moderate departures from normality, especially in large samples due to the Central Limit Theorem.
- Inference on predicted values, Y_{pred}, is more sensitive to normality violations.

When we assume a **normal error model**, the response variable Y_i is modeled as:

$$Y_i \sim N(\beta_0 + \beta_1 X_i, \sigma^2),$$

where:

- $\beta_0 + \beta_1 X_i$: Mean response
- σ^2: Variance of the errors

Under this model, the sampling distributions of the OLS estimators, b_0 and b_1, can be derived.

5.1.1.6.1 Inference for β_1 (Slope) Under the normal error model:

1. **Sampling Distribution of b_1:**

$$b_1 \sim N\left(\beta_1, \frac{\sigma^2}{\sum_{i=1}^n (X_i - \bar{X})^2}\right).$$

   This indicates that b_1 is an unbiased estimator of β_1 with variance proportional to σ^2.

2. **Test Statistic:**

$$t = \frac{b_1 - \beta_1}{s(b_1)} \sim t_{n-2},$$

   where $s(b_1)$ is the standard error of b_1:

$$s(b_1) = \sqrt{\frac{MSE}{\sum_{i=1}^n (X_i - \bar{X})^2}}.$$

3. **Confidence Interval:**

   A $(1-\alpha)100\%$ confidence interval for β_1 is:

$$b_1 \pm t_{1-\alpha/2;n-2} \cdot s(b_1).$$

5.1.1.6.2 Inference for β_0 (Intercept)

1. **Sampling Distribution of b_0:**

   Under the normal error model, the sampling distribution of b_0 is:

$$b_0 \sim N\left(\beta_0, \sigma^2\left(\frac{1}{n} + \frac{\bar{X}^2}{\sum_{i=1}^n (X_i - \bar{X})^2}\right)\right).$$

2. **Test Statistic:**

$$t = \frac{b_0 - \beta_0}{s(b_0)} \sim t_{n-2},$$

where $s(b_0)$ is the standard error of b_0:

$$s(b_0) = \sqrt{MSE\left(\frac{1}{n} + \frac{\bar{X}^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}\right)}.$$

3. **Confidence Interval:**

A $(1-\alpha)100\%$ confidence interval for β_0 is:

$$b_0 \pm t_{1-\alpha/2;n-2} \cdot s(b_0).$$

5.1.1.6.3 Mean Response In regression, we often estimate the mean response of the dependent variable Y for a given level of the predictor variable X, denoted as X_h. This estimation provides a predicted average outcome for a specific value of X based on the fitted regression model.

- Let X_h represent the level of X for which we want to estimate the mean response.
- The mean response when $X = X_h$ is denoted as $E(Y_h)$.
- A point estimator for $E(Y_h)$ is $\hat{Y}_h$, which is the predicted value from the regression model:

$$\hat{Y}_h = b_0 + b_1 X_h.$$

The estimator $\hat{Y}_h$ is unbiased because its expected value equals the true mean response $E(Y_h)$:

$$\begin{aligned} E(\hat{Y}_h) &= E(b_0 + b_1 X_h) \\ &= \beta_0 + \beta_1 X_h \\ &= E(Y_h). \end{aligned}$$

Thus, $\hat{Y}_h$ provides a reliable estimate of the mean response at X_h.

The variance of $\hat{Y}_h$ reflects the uncertainty in the estimate of the mean response:

$$\begin{aligned}
\text{Var}(\hat{Y}_h) &= \text{Var}(b_0 + b_1 X_h) \quad \text{(definition of } \hat{Y}_h\text{)} \\
&= \text{Var}((\bar{Y} - b_1\bar{X}) + b_1 X_h) \quad \text{(since } b_0 = \bar{Y} - b_1\bar{X}\text{)} \\
&= \text{Var}(\bar{Y} + b_1(X_h - \bar{X})) \quad \text{(factor out } b_1\text{)} \\
&= \text{Var}(\bar{Y} + b_1(X_h - \bar{X})) \\
&= \text{Var}(\bar{Y}) + (X_h - \bar{X})^2\text{Var}(b_1) + 2(X_h - \bar{X})\text{Cov}(\bar{Y}, b_1).
\end{aligned}$$

Since $\text{Cov}(\bar{Y}, b_1) = 0$ (due to the independence of the errors, ϵ_i), the variance simplifies to:

$$\text{Var}(\hat{Y}_h) = \frac{\sigma^2}{n} + (X_h - \bar{X})^2 \frac{\sigma^2}{\sum_{i=1}^n (X_i - \bar{X})^2}.$$

This can also be expressed as:

$$\text{Var}(\hat{Y}_h) = \sigma^2 \left(\frac{1}{n} + \frac{(X_h - \bar{X})^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \right).$$

To estimate the variance of $\hat{Y}_h$, we replace σ^2 with MSE, the mean squared error from the regression:

$$s^2(\hat{Y}_h) = MSE \left(\frac{1}{n} + \frac{(X_h - \bar{X})^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \right).$$

Under the normal error model, the sampling distribution of $\hat{Y}_h$ is:

$$\begin{aligned}
\hat{Y}_h &\sim N\left(E(Y_h), \text{Var}(\hat{Y}_h)\right), \\
\frac{\hat{Y}_h - E(Y_h)}{s(\hat{Y}_h)} &\sim t_{n-2}.
\end{aligned}$$

This result follows because $\hat{Y}_h$ is a linear combination of normally distributed random variables, and its variance is estimated using $s^2(\hat{Y}_h)$.

A $100(1-\alpha)\%$ confidence interval for the mean response $E(Y_h)$ is given by:

$$\hat{Y}_h \pm t_{1-\alpha/2;n-2} \cdot s(\hat{Y}_h),$$

where:

- $\widehat{Y}_h$: Point estimate of the mean response,
- $s(\widehat{Y}_h)$: Estimated standard error of the mean response,
- $t_{1-\alpha/2;n-2}$: Critical value from the t-distribution with $n-2$ degrees of freedom.

5.1.1.6.4 Prediction of a New Observation When analyzing regression results, it is important to distinguish between:

1. **Estimating the mean response** at a particular value of X.
2. **Predicting an individual outcome** for a particular value of X.

Mean Response vs. Individual Outcome

- **Same Point Estimate**
  The formula for both the estimated mean response and the predicted individual outcome at $X = X_h$ is identical:

$$\widehat{Y}_{pred} = \widehat{Y}_h = b_0 + b_1 X_h.$$

- **Different Variance**
  Although the point estimates are the same, the level of uncertainty differs. When predicting an individual outcome, we must consider not only the uncertainty in estimating the mean response ($\widehat{Y}_h$) but also the additional random variation within the distribution of Y.

Therefore, **prediction intervals** (for individual outcomes) account for more uncertainty and are consequently wider than **confidence intervals** (for the mean response).

To predict an individual outcome for a given X_h, we combine the mean response with the random error:

$$Y_{pred} = \beta_0 + \beta_1 X_h + \epsilon.$$

Using the least squares predictor:

$$\widehat{Y}_{pred} = b_0 + b_1 X_h,$$

since $E(\epsilon) = 0$.

The variance of the predicted value for a new observation, Y_{pred}, includes both:

1. Variance of the estimated mean response:

$$\sigma^2 \left(\frac{1}{n} + \frac{(X_h - \bar{X})^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \right),$$

2. Variance of the error term, ϵ, which is σ^2.

Thus, the total variance is:

$$\begin{aligned}
\text{Var}(Y_{pred}) &= \text{Var}(b_0 + b_1 X_h + \epsilon) \\
&= \text{Var}(b_0 + b_1 X_h) + \text{Var}(\epsilon) \\
&= \sigma^2 \left(\frac{1}{n} + \frac{(X_h - \bar{X})^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \right) + \sigma^2 \\
&= \sigma^2 \left(1 + \frac{1}{n} + \frac{(X_h - \bar{X})^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \right).
\end{aligned}$$

We estimate the variance of the prediction using MSE, the mean squared error:

$$s^2(pred) = MSE \left(1 + \frac{1}{n} + \frac{(X_h - \bar{X})^2}{\sum_{i=1}^n (X_i - \bar{X})^2} \right).$$

Under the normal error model, the standardized predicted value follows a t-distribution with $n-2$ degrees of freedom:

$$\frac{Y_{pred} - \hat{Y}_h}{s(pred)} \sim t_{n-2}.$$

A $100(1-\alpha)\%$ prediction interval for Y_{pred} is:

$$\hat{Y}_{pred} \pm t_{1-\alpha/2;n-2} \cdot s(pred).$$

5.1.1.6.5 Confidence Band In regression analysis, we often want to evaluate the uncertainty around the entire regression line, not just at a single value of the predictor variable X. This is achieved using a **confidence band**, which provides a confidence interval for the mean response, $E(Y) = \beta_0 + \beta_1 X$, over the entire range of X values.

The Working-Hotelling confidence band is a method to construct simultaneous confidence intervals for the regression line. For a given X_h, the confidence band is expressed as:

$$\hat{Y}_h \pm W s(\hat{Y}_h),$$

where:

- $W^2 = 2F_{1-\alpha;2,n-2}$,
  - $F_{1-\alpha;2,n-2}$ is the critical value from the F-distribution with 2 and $n-2$ degrees of freedom.
- $s(\hat{Y}_h)$ is the standard error of the estimated mean response at X_h:

$$s^2(\hat{Y}_h) = MSE\left(\frac{1}{n} + \frac{(X_h - \bar{X})^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}\right).$$

Key Properties of the Confidence Band

1. **Width of the Interval:**
   - The width of the confidence band changes with X_h because $s(\hat{Y}_h)$ depends on how far X_h is from the mean of X ($\bar{X}$).
   - The interval is narrowest at $X = \bar{X}$, where the variance of the estimated mean response is minimized.
2. **Shape of the Band:**
   - The boundaries of the confidence band form a hyperbolic shape around the regression line.
   - This reflects the increasing uncertainty in the mean response as X_h moves farther from $\bar{X}$.
3. **Simultaneous Coverage:**
   - The Working-Hotelling band ensures that the true regression line $E(Y) = \beta_0 + \beta_1 X$ lies within the band across all values of X with a specified confidence level (e.g., 95%).

5.1.1.7 Analysis of Variance (ANOVA) in Regression

ANOVA in regression decomposes the total variability in the response variable (Y) into components attributed to the regression model and residual error. In the context of regression, ANOVA provides a mechanism to assess the fit of the model and test hypotheses about the relationship between X and Y.

The **corrected Total Sum of Squares (SSTO)** quantifies the total variation in Y:

$$SSTO = \sum_{i=1}^{n}(Y_i - \bar{Y})^2,$$

where $\bar{Y}$ is the mean of the response variable. The term "corrected" refers to the fact that the sum of squares is calculated relative to the mean (i.e., the uncorrected total sum of squares is given by $\sum Y_i^2$)

Using the fitted regression model $\hat{Y}_i = b_0 + b_1 X_i$, we estimate the conditional mean of Y at X_i. The total sum of squares can be decomposed as:

$$\begin{aligned}
\sum_{i=1}^{n}(Y_i - \bar{Y})^2 &= \sum_{i=1}^{n}(Y_i - \hat{Y}_i + \hat{Y}_i - \bar{Y})^2 \\
&= \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2 + 2\sum_{i=1}^{n}(Y_i - \hat{Y}_i)(\hat{Y}_i - \bar{Y}) \\
&= \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2
\end{aligned}$$

- The cross-product term is zero, as shown below.
- This decomposition simplifies to:

$$SSTO = SSE + SSR,$$

where:

- $SSE = \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$: Error Sum of Squares (variation unexplained by the model).

- $SSR = \sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2$: Regression Sum of Squares (variation explained by the model), which measure how the conditional mean varies about a central value.

Degrees of freedom are partitioned as:

$$\begin{gathered}
SSTO = SSR + SSE \\
(n-1) = (1) + (n-2)
\end{gathered}$$

To confirm that the cross-product term is zero:

$$
\begin{aligned}
\sum_{i=1}^{n}(Y_i - \hat{Y}_i)(\hat{Y}_i - \bar{Y}) &= \sum_{i=1}^{n}(Y_i - \bar{Y} - b_1(X_i - \bar{X}))(\bar{Y} + b_1(X_i - \bar{X}) - \bar{Y}) \quad \text{(Expand } Y_i - \hat{Y}_i \text{ and } \hat{Y}_i - \bar{Y}\text{)} \\
&= \sum_{i=1}^{n}(Y_i - \bar{Y} - b_1(X_i - \bar{X}))(b_1(X_i - \bar{X})) \\
&= b_1 \sum_{i=1}^{n}(Y_i - \bar{Y})(X_i - \bar{X}) - b_1^2 \sum_{i=1}^{n}(X_i - \bar{X})^2 \quad \text{(Distribute terms in the product)} \\
&= b_1 \frac{\sum_{i=1}^{n}(Y_i - \bar{Y})(X_i - \bar{X})}{\sum_{i=1}^{n}(X_i - \bar{X})^2} \sum_{i=1}^{n}(X_i - \bar{X})^2 - b_1^2 \sum_{i=1}^{n}(X_i - \bar{X})^2 \quad \text{(Substitute } b_1 \text{ defin} \\
&= b_1^2 \sum_{i=1}^{n}(X_i - \bar{X})^2 - b_1^2 \sum_{i=1}^{n}(X_i - \bar{X})^2 \\
&= 0
\end{aligned}
$$

The ANOVA table summarizes the partitioning of variability:

Source of Variation	Sum of Squares	df	Mean Square	F Statistic
Regression (Model)	SSR	1	$MSR = \frac{SSR}{1}$	$F = \frac{MSR}{MSE}$
Error	SSE	$n-2$	$MSE = \frac{SSE}{n-2}$	
Total (Corrected)	$SSTO$	$n-1$		

The expected values of the mean squares are:

$$
\begin{aligned}
E(MSE) &= \sigma^2, \\
E(MSR) &= \sigma^2 + \beta_1^2 \sum_{i=1}^{n}(X_i - \bar{X})^2.
\end{aligned}
$$

- **If** $\beta_1 = 0$:
  - The regression model does not explain any variation in Y beyond the mean, and $E(MSR) = E(MSE) = \sigma^2$.
  - This condition corresponds to the null hypothesis, $H_0 : \beta_1 = 0$.
- **If** $\beta_1 \neq 0$:
  - The regression model explains some variation in Y, and $E(MSR) > E(MSE)$.
  - The additional term $\beta_1^2 \sum_{i=1}^{n}(X_i - \bar{X})^2$ represents the variance explained by the predictor X.

The difference between $E(MSR)$ and $E(MSE)$ allows us to infer whether $\beta_1 \neq 0$ by comparing their ratio.

Assuming the errors ϵ_i are independent and identically distributed as $N(0, \sigma^2)$, and under the null hypothesis $H_0 : \beta_1 = 0$, we have:

1. The scaled MSE follows a chi-square distribution with $n-2$ degrees of freedom:

$$\frac{MSE}{\sigma^2} \sim \chi^2_{n-2}.$$

2. The scaled MSR follows a chi-square distribution with 1 degree of freedom:

$$\frac{MSR}{\sigma^2} \sim \chi^2_1.$$

3. These two chi-square random variables are independent.

The ratio of two independent chi-square random variables, scaled by their respective degrees of freedom, follows an F-distribution. Therefore, under H_0:

$$F = \frac{MSR}{MSE} \sim F_{1,n-2}.$$

The F-statistic tests whether the regression model provides a significant improvement over the null model (constant $E(Y)$).

The hypotheses for the F-test are:

- **Null Hypothesis** (H_0): $\beta_1 = 0$ (no relationship between X and Y).
- **Alternative Hypothesis** (H_a): $\beta_1 \neq 0$ (a significant relationship exists between X and Y).

The rejection rule for H_0 at significance level α is:

$$F > F_{1-\alpha;1,n-2},$$

where $F_{1-\alpha;1,n-2}$ is the critical value from the F-distribution with 1 and $n-2$ degrees of freedom.

1. **If** $F \leq F_{1-\alpha;1,n-2}$:
   - Fail to reject H_0. There is insufficient evidence to conclude that X significantly explains variation in Y.
2. **If** $F > F_{1-\alpha;1,n-2}$:
   - Reject H_0. There is significant evidence that X explains some of the variation in Y.

5.1.1.8 Coefficient of Determination (R^2)

The **Coefficient of Determination** (R^2) measures how well the linear regression model accounts for the variability in the response variable Y. It is defined as:

$$R^2 = \frac{SSR}{SSTO} = 1 - \frac{SSE}{SSTO},$$

where:

- SSR: Regression Sum of Squares (variation explained by the model).
- $SSTO$: Total Sum of Squares (total variation in Y about its mean).
- SSE: Error Sum of Squares (variation unexplained by the model).

Properties of R^2

1. **Range**:
$$0 \leq R^2 \leq 1.$$
   - $R^2 = 0$: The model explains none of the variability in Y (e.g., $\beta_1 = 0$).
   - $R^2 = 1$: The model explains all the variability in Y (perfect fit).

2. **Proportionate Reduction in Variance**: R^2 represents the proportionate reduction in the total variation of Y after fitting the model. It quantifies how much better the model predicts Y compared to simply using $\bar{Y}$.

3. **Potential Misinterpretation**: It is not really correct to say R^2 is the "variation in Y explained by X." The term "variation explained" assumes a causative or deterministic explanation, which is not always correct. For example:
   - R^2 shows how much variance in Y is accounted for by the regression model, but it does not imply causation.
   - In cases with confounding variables or spurious correlations, R^2 can still be high, even if there's no direct causal link between X and Y.

For simple linear regression, R^2 is the square of the Pearson correlation coefficient, r:

$$R^2 = (r)^2,$$

where:

- $r = \text{corr}(X, Y)$ is the sample correlation coefficient.

The relationship between b_1 (the slope of the regression line) and r is given by:

$$b_1 = \left(\frac{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2} \right)^{1/2}.$$

Additionally, r can be expressed as:

$$r = \frac{s_y}{s_x} \cdot r,$$

where s_y and s_x are the sample standard deviations of Y and X, respectively.

5.1.1.9 Lack of Fit in Regression

The **lack of fit** test evaluates whether the chosen regression model adequately captures the relationship between the predictor variable X and the response variable Y. When there are repeated observations at specific values of X, we can partition the Error Sum of Squares (SSE) into two components:

1. **Pure Error**
2. **Lack of Fit**.

Given the observations:

- Y_{ij}: The j-th replicate for the i-th distinct value of X,
  - $Y_{11}, Y_{21}, \dots, Y_{n_1,1}$: n_1 repeated observations of X_1
  - $Y_{1c}, Y_{2c}, \dots, Y_{n_c,c}$: n_c repeated observations of X_c
- $\bar{Y}_j$: The mean response for replicates at X_j,
- $\hat{Y}_{ij}$: The predicted value from the regression model for X_j,

the Error Sum of Squares (SSE) can be decomposed as:

$$\begin{aligned}
\sum_i \sum_j (Y_{ij} - \hat{Y}_{ij})^2 &= \sum_i \sum_j (Y_{ij} - \bar{Y}_j + \bar{Y}_j - \hat{Y}_{ij})^2 \\
&= \sum_i \sum_j (Y_{ij} - \bar{Y}_j)^2 + \sum_j n_j (\bar{Y}_j - \hat{Y}_{ij})^2 + \text{cross product term} \\
&= \sum_i \sum_j (Y_{ij} - \bar{Y}_j)^2 + \sum_j n_j (\bar{Y}_j - \hat{Y}_{ij})^2
\end{aligned}$$

- The **cross product term** is zero because the deviations within replicates and the deviations between replicates are orthogonal.
- This simplifies to:

$$SSE = SSPE + SSLF,$$

where:

- $SSPE$ (Pure Error Sum of Squares): Variation within replicates for the same X_j, reflecting natural variability in the response.
  - Degrees of freedom: $df_{pe} = n - c$, where n is the total number of observations, and c is the number of distinct X values.
- $SSLF$ (Lack of Fit Sum of Squares): Variation between the replicate means $\bar{Y}_j$ and the model-predicted values $\hat{Y}_{ij}$. If SSLF is large, it suggests the model may not adequately describe the relationship between X and Y.
  - Degrees of freedom: $df_{lf} = c-2$, where 2 accounts for the parameters in the linear regression model (β_0 and β_1).

- **Mean Square for Pure Error (MSPE):**

$$MSPE = \frac{SSPE}{df_{pe}} = \frac{SSPE}{n-c}.$$

- **Mean Square for Lack of Fit (MSLF):**

$$MSLF = \frac{SSLF}{df_{lf}} = \frac{SSLF}{c-2}.$$

5.1.1.9.1 The F-Test for Lack of Fit The **F-test for lack of fit** evaluates whether the chosen regression model adequately captures the relationship between the predictor variable X and the response variable Y. Specifically, it tests whether any systematic deviations from the model exist that are not accounted for by random error.

- **Null Hypothesis (H_0):**
  The regression model is adequate:

$$H_0 : Y_{ij} = \beta_0 + \beta_1 X_i + \epsilon_{ij}, \quad \epsilon_{ij} \sim \text{i.i.d. } N(0, \sigma^2).$$

- **Alternative Hypothesis (H_a):**
  The regression model is not adequate and includes an additional function $f(X_i, Z_1, \dots)$ to account for the lack of fit:

$$H_a : Y_{ij} = \alpha_0 + \alpha_1 X_i + f(X_i, Z_1, \dots) + \epsilon^*_{ij}, \quad \epsilon^*_{ij} \sim \text{i.i.d. } N(0, \sigma^2).$$

Expected Mean Squares

- The expected Mean Square for Pure Error (MSPE) is the same under both H_0 and H_a:

$$E(MSPE) = \sigma^2.$$

- The expected Mean Square for Lack of Fit (MSLF) depends on whether H_0 is true:
  - Under H_0 (model is adequate):
$$E(MSLF) = \sigma^2.$$
  - Under H_a (model is not adequate):
$$E(MSLF) = \sigma^2 + \frac{\sum n_j f(X_i, Z_1, \dots)^2}{n-2}.$$

The test statistic for the lack-of-fit test is:

$$F = \frac{MSLF}{MSPE},$$

where:

- $MSLF = \frac{SSLF}{c-2}$,
  and $SSLF$ is the Lack of Fit Sum of Squares.
- $MSPE = \frac{SSPE}{n-c}$,
  and $SSPE$ is the Pure Error Sum of Squares.

Under H_0, the F-statistic follows an F-distribution:

$$F \sim F_{c-2,n-c}.$$

Decision Rule

- Reject H_0 at significance level α if:
$$F > F_{1-\alpha;c-2,n-c}.$$
- **Failing to reject** H_0:
  - Indicates that there is no evidence of lack of fit.
  - Does not imply the model is "true," but it suggests that the model provides a reasonable approximation to the true relationship.

To summarize, when repeat observations exist at some levels of X, the **Error Sum of Squares (SSE)** can be further partitioned into **Lack of Fit (SSLF)** and **Pure Error (SSPE)**. This leads to an extended ANOVA table:

Source of Variation	Sum of Squares	df	Mean Square	F Statistic
Regression	SSR	1	$MSR = \frac{SSR}{1}$	$F = \frac{MSR}{MSE}$
Error	SSE	$n-2$	$MSE = \frac{SSE}{n-2}$	
Lack of fit	SSLF	$c-2$	$MSLF = \frac{SSLF}{c-2}$	$F = \frac{MSLF}{MSPE}$
Pure Error	SSPE	$n-c$	$MSPE = \frac{SSPE}{n-c}$	
Total (Corrected)	SSTO	$n-1$		

Repeat observations have important implications for the coefficient of determination (R^2):

1. R^2 Can't Attain 1 with Repeat Observations:
   - With repeat observations, SSE (Error Sum of Squares) cannot be reduced to 0 because $SSPE > 0$ (variability within replicates).
2. **Maximum** R^2:
   - The maximum attainable R^2 in the presence of repeat observations is:

$$R^2_{\max} = \frac{SSTO - SSPE}{SSTO}.$$

3. **Importance of Repeat Observations:**
   - Not all levels of X need repeat observations, but their presence enables the separation of pure error from lack of fit, making the F-test for lack of fit possible.

Estimation of σ^2 with Repeat Observations

1. **Use of MSE:**
   - When H_0 is appropriate (the model fits well), MSE is typically used as the estimate of σ^2 instead of $MSPE$ because it has more degrees of freedom and provides a more reliable estimate.
2. **Pooling Estimates:**
   - In practice, MSE and $MSPE$ may be pooled if H_0 holds, resulting in a more precise estimate of σ^2.

5.1.1.10 Joint Inference for Regression Parameters

Joint inference considers the simultaneous coverage of confidence intervals for multiple regression parameters, such as β_0 (intercept) and β_1 (slope). Ensur-

ing adequate confidence for both parameters together requires adjustments to maintain the desired family-wise confidence level.

Let:

- $\bar{A}_1$: The event that the confidence interval for β_0 covers its true value.
- $\bar{A}_2$: The event that the confidence interval for β_1 covers its true value.

The individual confidence levels are:

$$P(\bar{A}_1) = 1 - \alpha,$$
$$P(\bar{A}_2) = 1 - \alpha.$$

The joint confidence coefficient, $P(\bar{A}_1 \cap \bar{A}_2)$, is:

$$\begin{aligned} P(\bar{A}_1 \cap \bar{A}_2) &= 1 - P(\bar{A}_1 \cup \bar{A}_2), \\ &= 1 - P(A_1) - P(A_2) + P(A_1 \cap A_2), \\ &\geq 1 - P(A_1) - P(A_2), \\ &= 1 - 2\alpha. \end{aligned}$$

This means that if α is the significance level for each parameter, the joint confidence coefficient is at least $1-2\alpha$. This inequality is known as the **Bonferroni Inequality**.

Bonferroni Confidence Intervals

To ensure a desired joint confidence level of $(1-\alpha)$ for both β_0 and β_1, the Bonferroni method adjusts the confidence level for each parameter by dividing α by the number of parameters. For two parameters:

1. The confidence level for each parameter is $(1-\alpha/2)$.
2. The resulting **Bonferroni-adjusted confidence intervals** are:

$$b_0 \pm B \cdot s(b_0),$$
$$b_1 \pm B \cdot s(b_1),$$

   where $B = t_{1-\alpha/4;n-2}$ is the critical value from the t-distribution with $n-2$ degrees of freedom.

Interpretation of Bonferroni Confidence Intervals

1. **Coverage Probability:**
   - If repeated samples were taken, $(1-\alpha)100\%$ of the joint intervals would contain the true values of (β_0, β_1).

- This implies that $\alpha \times 100\%$ of the samples would miss at least one of the true parameter values.

2. **Conservatism:**
   - The Bonferroni method ensures the family-wise confidence level but is **conservative**. The actual joint confidence level is often higher than $(1-\alpha)100\%$.
   - This conservatism reduces statistical power.

```r
# Load necessary libraries
library(ggplot2)
library(MASS)

# Set seed for reproducibility
set.seed(123)

# Generate synthetic data
n <- 100  # Number of observations
x <- rnorm(n, mean = 0, sd = 1)  # Predictor
beta_0 <- 2  # True intercept
beta_1 <- 3  # True slope
sigma <- 1  # Standard deviation of error
y <-
    beta_0 + beta_1 * x + rnorm(n, mean = 0, sd = sigma)  # Response

# Fit linear model
model <- lm(y ~ x)
summary(model)
#>
#> Call:
#> lm(formula = y ~ x)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -1.9073 -0.6835 -0.0875  0.5806  3.2904
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  1.89720    0.09755   19.45   <2e-16 ***
#> x            2.94753    0.10688   27.58   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.9707 on 98 degrees of freedom
#> Multiple R-squared:  0.8859, Adjusted R-squared:  0.8847
#> F-statistic: 760.6 on 1 and 98 DF,  p-value: < 2.2e-16
```

```r
# Extract coefficients and standard errors
b0_hat <- coef(model)[1]
b1_hat <- coef(model)[2]
s_b0 <-
    summary(model)$coefficients[1, 2]  # Standard error of intercept
s_b1 <- summary(model)$coefficients[2, 2]  # Standard error of slope

# Desired confidence level
alpha <- 0.05  # Overall significance level

# Bonferroni correction
adjusted_alpha <- alpha / 2  # Adjusted alpha for each parameter

# Critical t-value for Bonferroni adjustment
t_crit <-
    qt(1 - adjusted_alpha, df = n - 2)  # n-2 degrees of freedom

# Bonferroni confidence intervals
ci_b0 <- c(b0_hat - t_crit * s_b0, b0_hat + t_crit * s_b0)
ci_b1 <- c(b1_hat - t_crit * s_b1, b1_hat + t_crit * s_b1)

# Print results
cat("Bonferroni Confidence Intervals:\n")
#> Bonferroni Confidence Intervals:
cat("Intercept (beta_0): [",
    round(ci_b0[1], 2),
    ",",
    round(ci_b0[2], 2),
    "]\n")
#> Intercept (beta_0): [ 1.7 , 2.09 ]
cat("Slope (beta_1): [",
    round(ci_b1[1], 2),
    ",",
    round(ci_b1[2], 2),
    "]\n")
#> Slope (beta_1): [ 2.74 , 3.16 ]

# Calculate the covariance matrix of coefficients
cov_matrix <- vcov(model)

# Generate points for confidence ellipse
ellipse_points <-
    MASS::mvrnorm(n = 1000,
                  mu = coef(model),
                  Sigma = cov_matrix)
```

```r
# Convert to data frame for plotting
ellipse_df <- as.data.frame(ellipse_points)
colnames(ellipse_df) <- c("beta_0", "beta_1")

# Plot confidence intervals and ellipse
p <- ggplot() +
    # Confidence ellipse
    geom_point(
        data = ellipse_df,
        aes(x = beta_0, y = beta_1),
        alpha = 0.1,
        color = "grey"
    ) +
    # Point estimate
    geom_point(aes(x = b0_hat, y = b1_hat),
               color = "red",
               size = 3) +
    # Bonferroni confidence intervals
    geom_errorbar(aes(x = b0_hat, ymin = ci_b1[1], ymax = ci_b1[2]),
                  width = 0.1,
                  color = "blue") +
    geom_errorbarh(aes(y = b1_hat, xmin = ci_b0[1], xmax = ci_b0[2]),
                   height = 0.1,
                   color = "blue") +
    labs(title = "Bonferroni Confidence Intervals and Joint Confidence Region",
         x = "Intercept (beta_0)",
         y = "Slope (beta_1)") +
    theme_minimal()

print(p)
```

1. The red point represents the estimated coefficients (b0_hat, b1_hat).
2. The blue lines represent the Bonferroni-adjusted confidence intervals for beta_0 and beta_1.
3. The grey points represent the joint confidence region based on the covariance matrix of coefficients.
4. The Bonferroni intervals ensure family-wise confidence level but are conservative.
5. Simulation results demonstrate how often the true values are captured in the intervals when repeated samples are drawn.

Notes:

1. Conservatism of Bonferroni Intervals
   - The **Bonferroni interval is conservative**:
     - The joint confidence level is a lower bound, ensuring family-wise coverage of at least $(1-\alpha)100\%$.
     - This conservatism results in wider intervals, reducing the statistical power of the test.
   - Adjustments for Conservatism:
     - Practitioners often choose a larger α (e.g., $\alpha = 0.1$) to reduce the width of the intervals in Bonferroni joint tests.
     - A higher α allows for a better balance between confidence and precision, especially for exploratory analyses.
2. **Extending Bonferroni to Multiple Parameters**: The Bonferroni method is not limited to two parameters. For testing g parameters, such as $\beta_0, \beta_1, \dots, \beta_{g-1}$:
   - **Adjusted Confidence Level for Each Parameter:**
     - The confidence level for each individual parameter is $(1-\alpha/g)$.

- **Critical** t-Value:
  - For two-sided intervals, the critical value for each parameter is:

$$t_{1-\frac{\alpha}{2g};n-p},$$

    where p is the total number of parameters in the regression model.
- **Example:**
  - If $\alpha = 0.05$ and $g = 10$, each individual confidence interval is constructed at the:

$$(1 - \frac{0.05}{10}) = 99.5\% \text{ confidence level.}$$

  - This corresponds to using $t_{1-\frac{0.005}{2};n-p}$ in the formula for the confidence intervals.

3. **Limitations** for Large g
   - **Wide Intervals:**
     - As g increases, the intervals become excessively wide, often leading to reduced usefulness in practical applications.
     - This issue stems from the conservatism of the Bonferroni method, which prioritizes family-wise error control.
   - **Suitability for Small** g:
     - The Bonferroni procedure works well when g is relatively small (e.g., $g \leq 5$).
     - For larger g, alternative methods (discussed below) are more efficient.
4. Correlation Between Parameters: Correlation of b_0 and b_1:
   - The estimated regression coefficients b_0 and b_1 are often correlated:
     - **Negative correlation** if $\bar{X} > 0$.
     - **Positive correlation** if $\bar{X} < 0$.
   - This correlation can complicate joint inference but does not affect the validity of Bonferroni-adjusted intervals.
5. Alternatives to Bonferroni

Several alternative procedures provide more precise joint inference, especially for larger g:

1. **Scheffé's Method:**
   - Constructs simultaneous confidence regions for all possible linear combinations of parameters.
   - Suitable for exploratory analyses but may result in even wider intervals than Bonferroni.
2. **Tukey's Honest Significant Difference:**
   - Designed for pairwise comparisons in ANOVA but can be adapted for regression parameters.
3. **Holm's Step-Down Procedure:**
   - A sequential testing procedure that is less conservative than Bonferroni while still controlling the family-wise error rate.

4. **Likelihood Ratio Tests:**
   - Construct joint confidence regions based on the likelihood function, offering more precision for large g.

5.1.1.11 Assumptions of Linear Regression

To ensure valid inference and reliable predictions in linear regression, the following assumptions must hold. We'll cover them in depth in the next section.

Assumption	Description
Linearity	Linear relationship between predictors and response.
Independence of Errors	Errors are independent (important in time-series/clustering).
Homoscedasticity	Constant variance of residuals across predictors.
Normality of Errors	Residuals are normally distributed.
No Multicollinearity	Predictors are not highly correlated.
No Outliers/Leverage Points	No undue influence from outliers or high-leverage points.
Exogeneity	Predictors are uncorrelated with the error term (no endogeneity).
Full Rank	Predictors are linearly independent (no perfect multicollinearity).

5.1.1.12 Diagnostics for Model Assumptions

Constant Variance

- To check homoscedasticity:
  - Plot residuals vs. fitted values or residuals vs. predictors.
  - Look for patterns or a funnel-shaped spread indicating heteroscedasticity.

Outliers

- Detect outliers using:
  - Residuals vs. predictors plot.
  - Box plots.
  - Stem-and-leaf plots.
  - Scatter plots.

Standardized Residuals:

Residuals can be standardized to have unit variance, known as **studentized residuals**:

$$r_i = \frac{e_i}{s(e_i)}.$$

Semi-Studentized Residuals:

A simplified standardization using the mean squared error (MSE):

$$e_i^* = \frac{e_i}{\sqrt{MSE}}.$$

Non-Independent Error Terms

- To detect non-independence:
  - Plot residuals vs. time for time-series data.
  - Residuals e_i are not independent because they depend on $\hat{Y}_i$, which is derived from the same regression function.
- Detect dependency by plotting the residual for the i-th response vs. the $(i-1)$-th.

Non-Normality of Error Terms

- To assess normality:
  - Plot distribution of residuals.
  - Create box plots, stem-and-leaf plots, or normal probability plots.
- Issues such as an incorrect regression function or non-constant error variance can distort the residual distribution.
- Normality tests require relatively large sample sizes to detect deviations.

Normality of Residuals

- Use tests based on the empirical cumulative distribution function (ECDF) (check Normality Assessment)

Constancy of Error Variance

- Statistical tests for homoscedasticity:
  - **Brown-Forsythe Test (Modified Levene Test)**:
    - Robust against non-normality, examines the variance of residuals across levels of predictors.
  - **Breusch-Pagan Test (Cook-Weisberg Test)**:
    - Tests for heteroscedasticity by regressing squared residuals on predictors.

5.1.1.13 Remedial Measures for Violations of Assumptions

When the assumptions of simple linear regression are violated, appropriate remedial measures can be applied to address these issues. Below is a list of measures for specific deviations from the assumptions.

5.1.1.13.1 General Remedies

- Use more **complicated models** (e.g., non-linear models, generalized linear models).

- Apply **transformations** (see Variable Transformation) on X and/or Y to stabilize variance, linearize relationships, or normalize residuals. Note that transformations may not always yield "optimal" results.

5.1.1.13.2 Specific Remedies for Assumption Violations

Issue	Remedy	Explanation
Non-Linearity	- Apply transformations (e.g., log, square root).	Transformation of variables can help linearize the relationship between X and Y.
	- Use more complicated models (e.g., polynomial regression, splines).	Higher-order terms or non-linear models can capture non-linear relationships.
Non-Constant Error Variance	- Apply **Weighted Least Squares**.	WLS assigns weights to observations based on the inverse of their variance.
	- Use transformations (e.g., log, square root).	Transformations can stabilize error variance.
Correlated Errors	- Use serially correlated error models (e.g., ARIMA for time-series data).	Time-series models account for serial dependence in the errors.
Non-Normality of Errors	- Transform Y or use non-parametric methods.	Transformations can normalize residuals; non-parametric methods do not assume normality.
Omitted Variables	- Use multiple regression to include additional relevant predictors.	Adding relevant variables reduces omitted variable bias and improves model accuracy.
Outliers	- Apply robust estimation techniques (e.g., Huber regression, M-estimation).	Robust methods reduce the influence of outliers on parameter estimates.

5.1.1.13.3 Remedies in Detail

1. **Non-Linearity:**
   - Transformations: Apply transformations to the response variable Y or the predictor variable X. Common transformations include:
     - Logarithmic transformation: $Y' = \log(Y)$ or $X' = \log(X)$.
     - Polynomial terms: Include X^2, X^3, etc., to capture curvature.

- Alternative Models:
  - Polynomial regression or splines for flexibility in modeling non-linear relationships.

2. **Non-Constant Error Variance:**
   - Weighted Least Squares:
     - Assigns weights to observations inversely proportional to their variance.
   - Transformations:
     - Use a log or square root transformation to stabilize variance.
3. **Correlated Errors:**
   - For time-series data:
     - Use serially correlated error models such as AR(1) or ARIMA.
     - These models explicitly account for dependency in residuals over time.
4. **Non-Normality:**
   - Transformations:
     - Apply a transformation to Y (e.g., log or square root) to make the residuals approximately normal.
   - Non-parametric regression:
     - Methods like LOESS or Theil-Sen regression do not require the normality assumption.
5. **Omitted Variables:**
   - Introduce additional predictors:
     - Use multiple regression to include all relevant independent variables.
   - Check for multicollinearity when adding new variables.
6. **Outliers:**
   - Robust Regression:
     - Use methods such as Huber regression or M-estimation to reduce the impact of outliers on model coefficients.
   - Diagnostics:
     - Identify outliers using Cook's Distance, leverage statistics, or studentized residuals.

5.1.1.14 Transformations in Regression Analysis

Transformations involve modifying one or both variables to address issues such as non-linearity, non-constant variance, or non-normality. However, it's important to note that the properties of least-squares estimates apply to the **transformed model**, not the original variables.

When transforming the dependent variable Y, we fit the model as:

$$g(Y_i) = b_0 + b_1 X_i,$$

where $g(Y_i)$ is the transformed response. To interpret the regression results in

terms of the original Y, we need to **transform back**:

$$\hat{Y}_i = g^{-1}(b_0 + b_1 X_i).$$

Direct back-transformation of predictions can introduce **bias**. For example, in a log-transformed model:

$$\log(Y_i) = b_0 + b_1 X_i,$$

the unbiased back-transformed prediction of Y_i is:

$$\hat{Y}_i = \exp(b_0 + b_1 X_i + \frac{\sigma^2}{2}),$$

where $\frac{\sigma^2}{2}$ accounts for the bias correction due to the log transformation.

5.1.1.14.1 Box-Cox Family of Transformations The **Box-Cox transformation** is a versatile family of transformations defined as:

$$Y' = \begin{cases} \frac{Y^\lambda - 1}{\lambda}, & \text{if } \lambda \neq 0, \\ \ln(Y), & \text{if } \lambda = 0. \end{cases}$$

This transformation introduces a parameter λ that is estimated from the data. Common transformations include:

λ	Transformation Y'
2	Y^2
0.5	$\sqrt{Y}$
0	$\ln(Y)$
-0.5	$1/\sqrt{Y}$
-1	$1/Y$

Choosing the Transformation Parameter λ

The value of λ can be selected using one of the following methods:

1. **Trial and Error**:
   - Apply different transformations and compare the residual plots or model fit statistics (e.g., R^2 or AIC).

2. **Maximum Likelihood Estimation**:
   - Choose λ to maximize the likelihood function under the assumption of normally distributed errors.
3. **Numerical Search**:
   - Use computational optimization techniques to minimize the residual sum of squares (RSS) or another goodness-of-fit criterion.

```
# Install and load the necessary library
if (!require("MASS")) install.packages("MASS")
library(MASS)

# Fit a linear model
set.seed(123)
n <- 50
x <- rnorm(n, mean = 5, sd = 2)
y <- 3 + 2 * x + rnorm(n, mean = 0, sd = 2)
model <- lm(y ~ x)

# Apply Box-Cox Transformation
boxcox_result <- boxcox(model, lambda = seq(-2, 2, 0.1), plotit = TRUE)
```

```
# Find the optimal lambda
optimal_lambda <- boxcox_result$x[which.max(boxcox_result$y)]
cat("Optimal lambda for Box-Cox transformation:", optimal_lambda, "\n")
#> Optimal lambda for Box-Cox transformation: 0.8686869
```

Notes

1. **Benefits of Transformations**:
   - **Stabilize Variance**: Helps address heteroscedasticity.
   - **Linearize Relationships**: Useful for non-linear data.
   - **Normalize Residuals**: Addresses non-normality issues.
2. **Caveats**:
   - Interpretability: Transformed variables may complicate interpretation.
   - Over-Transformation: Excessive transformations can distort the relationship between variables.
3. **Applicability**:
   - Transformations are most effective for issues like non-linearity or non-constant variance. They are less effective for correcting independence violations or omitted variables.

5.1.1.14.2 Variance Stabilizing Transformations Variance stabilizing transformations are used when the standard deviation of the response variable depends on its mean. The **delta method**, which applies a Taylor series expansion, provides a systematic approach to find such transformations.

Given that the standard deviation of Y is a function of its mean:

$$\sigma = \sqrt{\text{var}(Y)} = f(\mu),$$

where $\mu = E(Y)$ and $f(\mu)$ is a smooth function of the mean, we aim to find a transformation $h(Y)$ such that the variance of the transformed variable $h(Y)$ is constant for all values of μ.

Expanding $h(Y)$ in a Taylor Expansion series around μ:

$$h(Y) = h(\mu) + h'(\mu)(Y - \mu) + \text{higher-order terms}.$$

Ignoring higher-order terms, the variance of $h(Y)$ can be approximated as:

$$\text{var}(h(Y)) = \text{var}(h(\mu) + h'(\mu)(Y - \mu)).$$

Since $h(\mu)$ is a constant:

$$\text{var}(h(Y)) = (h'(\mu))^2 \, \text{var}(Y).$$

Substituting $\text{var}(Y) = (f(\mu))^2$, we get:

$$\text{var}(h(Y)) = (h'(\mu))^2 (f(\mu))^2 .$$

To stabilize the variance (make it constant for all μ), we require:

$$(h'(\mu))^2 (f(\mu))^2 = \text{constant}.$$

Thus, the derivative of $h(\mu)$ must be proportional to the inverse of $f(\mu)$:

$$h'(\mu) \propto \frac{1}{f(\mu)}.$$

Integrating both sides gives:

$$h(\mu) = \int \frac{1}{f(\mu)} \, d\mu.$$

The specific form of $h(\mu)$ depends on the function $f(\mu)$, which describes the relationship between the standard deviation and the mean.

Examples of Variance Stabilizing Transformations

$f(\mu)$	**Transformation** $h(Y)$	**Purpose**
$\sqrt{\mu}$	$\int \frac{1}{\sqrt{\mu}} d\mu = 2\sqrt{Y}$	Stabilizes variance for Poisson data.
μ	$\int \frac{1}{\mu} d\mu = \ln(Y)$	Stabilizes variance for exponential or multiplicative models.
μ^2	$\int \frac{1}{\mu^2} d\mu = -\frac{1}{Y}$	Stabilizes variance for certain power law data.

Variance stabilizing transformations are particularly useful for:

1. **Poisson-distributed data**: Use $h(Y) = 2\sqrt{Y}$ to stabilize variance.
2. **Exponential or multiplicative models**: Use $h(Y) = \ln(Y)$ for stabilization.
3. **Power law relationships**: Use transformations like $h(Y) = Y^{-1}$ or other forms derived from $f(\mu)$.

Example: Variance Stabilizing Transformation for the Poisson Distribution

For a **Poisson distribution**, the variance of Y is equal to its mean:

$$\sigma^2 = \text{var}(Y) = E(Y) = \mu.$$

Thus, the standard deviation is:

$$\sigma = f(\mu) = \sqrt{\mu}.$$

Using the relationship for variance stabilizing transformations:

$$h'(\mu) \propto \frac{1}{f(\mu)} = \mu^{-0.5}.$$

Integrating $h'(\mu)$ gives the variance stabilizing transformation:

$$h(\mu) = \int \mu^{-0.5}\, d\mu = 2\sqrt{\mu}.$$

Hence, the variance stabilizing transformation is:

$$h(Y) = \sqrt{Y}.$$

This transformation is widely used in Poisson regression to stabilize the variance of the response variable.

```r
# Simulate Poisson data
set.seed(123)
n <- 500
x <- rnorm(n, mean = 5, sd = 2)
y <- rpois(n, lambda = exp(1 + 0.3 * x))  # Poisson-distributed Y

# Fit linear model without transformation
model_raw <- lm(y ~ x)

# Apply square root transformation
y_trans <- sqrt(y)
model_trans <- lm(y_trans ~ x)

# Compare residual plots
par(mfrow = c(2, 1))

# Residual plot for raw data
plot(
    fitted(model_raw),
    resid(model_raw),
    main = "Residuals: Raw Data",
    xlab = "Fitted Values",
    ylab = "Residuals"
)
```

```r
abline(h = 0, col = "red", lty = 2)

# Residual plot for transformed data
plot(
    fitted(model_trans),
    resid(model_trans),
    main = "Residuals: Transformed Data (sqrt(Y))",
    xlab = "Fitted Values",
    ylab = "Residuals"
)
abline(h = 0, col = "blue", lty = 2)
```

5.1.1.14.3 General Strategy When $f(\mu)$ Is Unknown If the relationship between $\text{var}(Y)$ and μ (i.e., $f(\mu)$) is unknown, the following steps can help:

1. **Trial and Error**:
   - Apply common transformations (e.g., $\log(Y)$, $\sqrt{Y}$) and examine residual plots.
   - Select the transformation that results in stabilized variance (residuals show no pattern in plots).
2. **Leverage Prior Research**:
   - Consult researchers or literature on similar experiments to determine the transformations typically used.
3. **Analyze Observations with the Same Predictor Value**:
   - If multiple observations Y_{ij} are available at the same X value:
     - Compute the mean $\bar{Y}_i$ and standard deviation s_i for each group.
     - Check if $s_i \propto \bar{Y}_i^{\lambda}$.

    * For example, assume:

$$s_i = a\bar{Y}_i^{\lambda}.$$

    * Taking the natural logarithm:

$$\ln(s_i) = \ln(a) + \lambda \ln(\bar{Y}_i).$$

  - Perform a regression of $\ln(s_i)$ on $\ln(\bar{Y}_i)$ to estimate λ and suggest the form of $f(\mu)$.

4. **Group Observations**:
   - If individual observations are sparse, try grouping similar observations by X values to compute $\bar{Y}_i$ and s_i for each group.

5.1.1.14.4 Common Transformations and Their Applications The table below summarizes common transformations used to stabilize variance under various conditions, along with their appropriate contexts and comments:

Transformation	**Situation**	**Comments**
$\sqrt{Y}$	$var(\epsilon_i) = k\,E(Y_i)$	For counts following a Poisson distribution.
$\sqrt{Y} + \sqrt{Y+1}$	$var(\epsilon_i) = k\,E(Y_i)$	Useful for small counts or datasets with zeros.
$\log(Y)$	$var(\epsilon_i) = k\,(E(Y_i))^2$	Appropriate for positive integers with a wide range.
$\log(Y+1)$	$var(\epsilon_i) = k\,(E(Y_i))^2$	Used when the data includes zero counts.
$1/Y$	$var(\epsilon_i) = k\,(E(Y_i))^4$	For responses mostly near zero, with occasional large values.
$\arcsin(\sqrt{Y})$	$var(\epsilon_i) = k\,E(Y_i)(1 - E(Y_i))$	Suitable for binomial proportions or percentage data.

1. **Choosing the Transformation**:
   - Start by identifying the relationship between the variance of the residuals ($var(\epsilon_i)$) and the mean of the response variable ($E(Y_i)$).
   - Select the transformation that matches the identified variance structure.
2. **Transformations for Zero Values**:
   - For data with zeros, transformations like $\sqrt{Y+1}$ or $\log(Y+1)$ can be used to avoid undefined values. But this will seriously jeopardize model assumption (Chen and Roth, 2024).
3. **Use in Regression Models**:
   - Apply these transformations to the dependent variable Y in the regression model.

- Always check residual plots to confirm that the transformation stabilizes variance and resolves non-linearity.

4. **Interpretation After Transformation**:
   - After transforming Y, interpret the results in terms of the transformed variable.
   - For practical interpretation, back-transform predictions and account for any associated bias.

5.1.2 Multiple Linear Regression

The geometry of least squares regression involves projecting the response vector $\mathbf{y}$ onto the space spanned by the columns of the design matrix $\mathbf{X}$. The fitted values $\hat{\mathbf{y}}$ can be expressed as:

$$\begin{aligned}\hat{\mathbf{y}} &= \mathbf{X}\mathbf{b} \\ &= \mathbf{X}(\mathbf{X}'\mathbf{X})^{-\mathbf{1}}\mathbf{X}'\mathbf{y} \\ &= \mathbf{H}\mathbf{y},\end{aligned}$$

where:

- $\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-\mathbf{1}}\mathbf{X}'$ is the projection operator (sometimes denoted as $\mathbf{P}$).
- $\hat{\mathbf{y}}$ is the projection of $\mathbf{y}$ onto the linear space spanned by the columns of $\mathbf{X}$ (the model space).

The dimension of the model space is equal to the rank of $\mathbf{X}$ (i.e., the number of linearly independent columns in $\mathbf{X}$).

Properties of the Projection Matrix $\mathbf{H}$

1. Symmetry:
   - The projection matrix $\mathbf{H}$ is symmetric:

$$\mathbf{H} = \mathbf{H}'.$$

2. Idempotence:
   - Applying $\mathbf{H}$ twice gives the same result:

$$\mathbf{H}\mathbf{H} = \mathbf{H}.$$

   Proof:

$$\begin{aligned}\mathbf{H}\mathbf{H} &= \mathbf{X}(\mathbf{X}'\mathbf{X})^{-\mathbf{1}}\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-\mathbf{1}}\mathbf{X}' \\ &= \mathbf{X}(\mathbf{X}'\mathbf{X})^{-\mathbf{1}}\mathbf{I}\mathbf{X}' \\ &= \mathbf{X}(\mathbf{X}'\mathbf{X})^{-\mathbf{1}}\mathbf{X}' \\ &= \mathbf{H}.\end{aligned}$$

3. Dimensionality:

- $\mathbf{H}$ is an $n \times n$ matrix (where n is the number of observations).
- The rank of $\mathbf{H}$ is equal to the rank of $\mathbf{X}$, which is typically the number of predictors (including the intercept).

4. Orthogonal Complement:
   - The matrix $(\mathbf{I} - \mathbf{H})$, where:

$$\mathbf{I} - \mathbf{H} = \mathbf{I} - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}',$$

     is also a projection operator.
   - It projects onto the orthogonal complement of the space spanned by the columns of $\mathbf{X}$ (i.e., the space orthogonal to the model space).
5. Orthogonality of Projections:
   - $\mathbf{H}$ and $(\mathbf{I} - \mathbf{H})$ are orthogonal:

$$\mathbf{H}(\mathbf{I} - \mathbf{H}) = \mathbf{0}.$$

   - Similarly:

$$(\mathbf{I} - \mathbf{H})\mathbf{H} = \mathbf{0}.$$

Intuition for $\mathbf{H}$ and $(\mathbf{I} - \mathbf{H})$

- $\mathbf{H}$: Projects $\mathbf{y}$ onto the model space, giving the fitted values $\hat{\mathbf{y}}$.
- $\mathbf{I} - \mathbf{H}$: Projects $\mathbf{y}$ onto the residual space, giving the residuals $\mathbf{e}$:

$$\mathbf{e} = (\mathbf{I} - \mathbf{H})\mathbf{y}.$$

- $\mathbf{H}$ and $(\mathbf{I} - \mathbf{H})$ divide the response vector $\mathbf{y}$ into two components:

$$\mathbf{y} = \hat{\mathbf{y}} + \mathbf{e}.$$

  - $\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}$ (fitted values).
  - $\mathbf{e} = (\mathbf{I} - \mathbf{H})\mathbf{y}$ (residuals).
- The properties of $\mathbf{H}$ (symmetry, idempotence, dimensionality) reflect its role as a linear transformation that projects vectors onto the model space.

This geometric perspective provides insight into the mechanics of least squares regression, particularly how the response variable $\mathbf{y}$ is decomposed into fitted values and residuals.

Similar to simple regression, the total sum of squares in multiple regression analysis can be partitioned into components corresponding to the regression (model fit) and the residuals (errors).

The uncorrected total sum of squares is:

$$\mathbf{y}'\mathbf{y} = \hat{\mathbf{y}}'\hat{\mathbf{y}} + \mathbf{e}'\mathbf{e},$$

where:

- $\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}$ (fitted values, projected onto the model space).
- $\mathbf{e} = (\mathbf{I} - \mathbf{H})\mathbf{y}$ (residuals, projected onto the orthogonal complement of the model space).

Expanding this using projection matrices:

$$\begin{aligned}\mathbf{y}'\mathbf{y} &= (\mathbf{H}\mathbf{y})'(\mathbf{H}\mathbf{y}) + ((\mathbf{I} - \mathbf{H})\mathbf{y})'((\mathbf{I} - \mathbf{H})\mathbf{y}) \\ &= \mathbf{y}'\mathbf{H}'\mathbf{H}\mathbf{y} + \mathbf{y}'(\mathbf{I} - \mathbf{H})'(\mathbf{I} - \mathbf{H})\mathbf{y} \\ &= \mathbf{y}'\mathbf{H}\mathbf{y} + \mathbf{y}'(\mathbf{I} - \mathbf{H})\mathbf{y}.\end{aligned}$$

This equation shows the partition of $\mathbf{y}'\mathbf{y}$ into components explained by the model ($\hat{\mathbf{y}}$) and the unexplained variation (residuals).

For the corrected total sum of squares, we adjust for the mean (using the projection matrix $\mathbf{H_1}$):

$$\mathbf{y}'(\mathbf{I} - \mathbf{H_1})\mathbf{y} = \mathbf{y}'(\mathbf{H} - \mathbf{H_1})\mathbf{y} + \mathbf{y}'(\mathbf{I} - \mathbf{H})\mathbf{y}.$$

Here:

- $\mathbf{H_1} = \frac{1}{n}\mathbf{J}$, where $\mathbf{J}$ is an $n \times n$ matrix of ones.
- $\mathbf{H} - \mathbf{H_1}$ projects onto the subspace explained by the predictors after centering.

Aspect	Uncorrected Total Sum of Squares ($\mathbf{y}'\mathbf{y}$)	Corrected Total Sum of Squares ($\mathbf{y}'(\mathbf{I} - \mathbf{H_1})\mathbf{y}$)
Definition	Total variation in y relative to the origin.	Total variation in y relative to its mean (centered data).
Adjustment	No adjustment for the mean of y.	Adjusts for the mean of y by centering it.
Equation	$\mathbf{y}'\mathbf{y} = \hat{\mathbf{y}}'\hat{\mathbf{y}} + \mathbf{e}'\mathbf{e}$	$\mathbf{y}'(\mathbf{I} - \mathbf{H_1})\mathbf{y} = \mathbf{y}'(\mathbf{H} - \mathbf{H_1})\mathbf{y} + \mathbf{y}'(\mathbf{I} - \mathbf{H})\mathbf{y}$
Projection Matrices	$\mathbf{H}$: Projects onto model space. $\mathbf{I} - \mathbf{H}$: Projects onto residuals.	$\mathbf{H_1} = \frac{1}{n}\mathbf{J}$: Adjusts for the mean. $\mathbf{H} - \mathbf{H_1}$: Projects onto predictors after centering. $\mathbf{I} - \mathbf{H}$: Projects onto residuals.
Interpretation	Includes variation due to the mean of y.	Focuses on variation in y around its mean.
Usage	Suitable for raw, uncentered data.	Common in regression and ANOVA to isolate variability explained by predictors.
Application	Measures total variability in y, including overall level (mean).	Measures variability explained by predictors relative to the mean.

Why the Correction Matters

- In ANOVA and regression, removing the contribution of the mean helps isolate the variability explained by predictors from the overall level of the response variable.
- Corrected sums of squares are more common when comparing models or computing R^2, which requires centering to ensure consistency in proportionate variance explained.

The corrected total sum of squares can be decomposed into the sum of squares for regression (SSR) and the sum of squares for error (SSE):

Source	SS	df	MS	F
Regression	$SSR = \mathbf{y}'(\mathbf{H} - \frac{\mathbf{1}}{\mathbf{n}}\mathbf{J})\mathbf{y}$	$p-1$	$MSR = SSR/(p-1)$	MSR/MSE
Error	$SSE = \mathbf{y}'(\mathbf{I} - \mathbf{H})\mathbf{y}$	$n-p$	$MSE = SSE/(n-p)$	
Total	$SST = \mathbf{y}'(\mathbf{I} - \mathbf{H_1})\mathbf{y}$	$n-1$		

Where:

- p: Number of parameters (including intercept).
- n: Number of observations.

Alternatively, the regression model can be expressed as:

$$\mathbf{Y} = \mathbf{X} + (\mathbf{Y} - \mathbf{X}),$$

where:

- $\hat{\mathbf{Y}} = \mathbf{X}$: Vector of fitted values (in the subspace spanned by $\mathbf{X}$).
- $\mathbf{e} = \mathbf{Y} - \mathbf{X}$: Vector of residuals (in the orthogonal complement of the subspace spanned by $\mathbf{X}$).
- $\mathbf{Y}$ is an $n \times 1$ vector in the n-dimensional space $\mathbb{R}^n$.
- $\mathbf{X}$ is an $n \times p$ full-rank matrix, with its columns generating a p-dimensional subspace of $\mathbb{R}^n$. Hence, any estimator $\mathbf{X}$ is also in this subspace.

In linear regression, the Ordinary Least Squares estimator $\hat{\beta}$ minimizes the squared Euclidean distance $\|\mathbf{Y} - \mathbf{X}\beta\|^2$ between the observed response vector $\mathbf{Y}$ and the fitted values $\mathbf{X}\beta$. This minimization corresponds to the orthogonal projection of $\mathbf{Y}$ onto the column space of $\mathbf{X}$.

We solve the optimization problem:

$$\min_{\beta} \|\mathbf{Y} - \mathbf{X}\beta\|^2.$$

The objective function can be expanded as:

$$\|\mathbf{Y} - \mathbf{X}\beta\|^2 = (\mathbf{Y} - \mathbf{X}\beta)^\top (\mathbf{Y} - \mathbf{X}\beta).$$

Perform the multiplication:

$$(\mathbf{Y} - \mathbf{X}\beta)^\top (\mathbf{Y} - \mathbf{X}\beta) = \mathbf{Y}^\top \mathbf{Y} - \mathbf{Y}^\top \mathbf{X}\beta - \beta^\top \mathbf{X}^\top \mathbf{Y} + \beta^\top (\mathbf{X}^\top \mathbf{X})\beta.$$

Since $\mathbf{Y}^\top \mathbf{X}\beta$ is a scalar, it equals $\beta^\top \mathbf{X}^\top \mathbf{Y}$. Therefore, the expanded expression becomes:

$$\|\mathbf{Y} - \mathbf{X}\beta\|^2 = \mathbf{Y}^\top \mathbf{Y} - 2\beta^\top \mathbf{X}^\top \mathbf{Y} + \beta^\top (\mathbf{X}^\top \mathbf{X})\beta.$$

To find the β that minimizes this expression, take the derivative with respect to β and set it to 0:

$$\frac{\partial}{\partial \beta} \Big[\mathbf{Y}^\top \mathbf{Y} - 2\beta^\top \mathbf{X}^\top \mathbf{Y} + \beta^\top (\mathbf{X}^\top \mathbf{X})\beta \Big] = 0.$$

Computing the gradient:

$$\frac{\partial}{\partial \beta} = -2\mathbf{X}^\top \mathbf{Y} + 2(\mathbf{X}^\top \mathbf{X})\beta.$$

Setting this to zero:

$$-2\mathbf{X}^\top \mathbf{Y} + 2\mathbf{X}^\top \mathbf{X}\beta = 0.$$

Simplify:

$$\mathbf{X}^\top \mathbf{X}\beta = \mathbf{X}^\top \mathbf{Y}.$$

If $\mathbf{X}^\top \mathbf{X}$ is invertible, the solution is:

$$\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}.$$

Orthogonal Projection Interpretation

The fitted values are:

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\beta}.$$

From the normal equations, $\mathbf{X}^\top(\mathbf{Y} - \mathbf{X}\hat{\beta}) = 0$, which implies that the residual vector $\mathbf{Y} - \hat{\mathbf{Y}}$ is orthogonal to every column of $\mathbf{X}$. Therefore:

- $\hat{\mathbf{Y}} = \mathbf{X}\hat{\beta}$ is the orthogonal projection of $\mathbf{Y}$ onto $\mathrm{Col}(\mathbf{X})$.
- $\mathbf{Y} - \mathbf{X}\hat{\beta}$ lies in the orthogonal complement of $\mathrm{Col}(\mathbf{X})$.

Pythagoras Decomposition

The geometric interpretation gives us the decomposition:

$$\mathbf{Y} = \mathbf{X}\hat{\beta} + (\mathbf{Y} - \mathbf{X}\hat{\beta}),$$

where:

- $\mathbf{X}\hat{\beta}$ is the projection of $\mathbf{Y}$ onto the column space of $\mathbf{X}$.
- $(\mathbf{Y} - \mathbf{X}\hat{\beta})$ is the residual vector, orthogonal to $\mathbf{X}\hat{\beta}$.

Since the two components are orthogonal, their squared norms satisfy:

$$\begin{aligned}
\|\mathbf{Y}\|^2 &= \mathbf{Y}^\top\mathbf{Y} && \text{(definition of norm squared)} \\
&= (\mathbf{Y} - \mathbf{X}\hat{\beta} + \mathbf{X}\hat{\beta})^\top(\mathbf{Y} - \mathbf{X}\hat{\beta} + \mathbf{X}\hat{\beta}) && \text{(add and subtract the same term } \mathbf{X}\hat{\beta}) \\
&= (\mathbf{Y} - \mathbf{X}\hat{\beta})^\top(\mathbf{Y} - \mathbf{X}\hat{\beta}) \;+\; 2\,(\mathbf{X}\hat{\beta})^\top(\mathbf{Y} - \mathbf{X}\hat{\beta}) \;+\; (\mathbf{X}\hat{\beta})^\top(\mathbf{X}\hat{\beta}) && \text{(expand } (a+b)^\top(a+b)) \\
&= \|\mathbf{Y} - \mathbf{X}\hat{\beta}\|^2 \;+\; 2\,(\mathbf{X}\hat{\beta})^\top(\mathbf{Y} - \mathbf{X}\hat{\beta}) \;+\; \|\mathbf{X}\hat{\beta}\|^2 && \text{(rewrite each quadratic form as a norm)} \\
&= \|\mathbf{Y} - \mathbf{X}\hat{\beta}\|^2 + \|\mathbf{X}\hat{\beta}\|^2 && \text{(use that } (\mathbf{X}\hat{\beta})^\top(\mathbf{Y} - \mathbf{X}\hat{\beta}) = 0\text{, i.e. orthogonali} \\
&= \|\mathbf{X}\hat{\beta}\|^2 \;+\; \|\mathbf{Y} - \mathbf{X}\hat{\beta}\|^2.
\end{aligned}$$

where the norm of a vector $\mathbf{a}$ in $\mathbb{R}^p$ is defined as:

$$\|\mathbf{a}\| = \sqrt{\mathbf{a}^\top\mathbf{a}} = \sqrt{\sum_{i=1}^{p} a_i^2}.$$

We are saying that $\mathbf{Y}$ is decomposed into two orthogonal components:

1. $\mathbf{X}\hat{\beta}$ (the projection onto Col($\mathbf{X}$)
   - $\|\mathbf{X}\hat{\beta}\|$ measures the part of $\mathbf{Y}$ explained by the model.
2. $\mathbf{Y} - \mathbf{X}\hat{\beta}$ (the residual lying in the orthogonal complement).
   - $\|\mathbf{Y} - \mathbf{X}\hat{\beta}\|$ measures the residual error.

This geometric interpretation (projection plus orthogonal remainder) is exactly why we call $\mathbf{X}\hat{\beta}$ the *orthogonal projection* of $\mathbf{Y}$ onto the column space of $\mathbf{X}$. This decomposition also underlies the analysis of variance (ANOVA) in regression.

The coefficient of multiple determination, denoted R^2, measures the proportion of the total variation in the response variable ($\mathbf{Y}$) that is explained by the regression model. It is defined as:

$$R^2 = \frac{SSR}{SSTO} = 1 - \frac{SSE}{SSTO},$$

where:

- SSR: Regression sum of squares (variation explained by the model).
- SSE: Error sum of squares (unexplained variation).
- $SSTO$: Total sum of squares (total variation in $\mathbf{Y}$).

The adjusted R^2 adjusts R^2 for the number of predictors in the model, penalizing for adding predictors that do not improve the model's fit substantially. It is defined as:

$$R_a^2 = 1 - \frac{SSE/(n-p)}{SSTO/(n-1)} = 1 - \frac{(n-1)SSE}{(n-p)SSTO},$$

where:

- n: Number of observations.
- p: Number of parameters (including the intercept).

Key Differences Between R^2 and R_a^2

Aspect	R^2	R_a^2
Behavior with Predictors	Always increases (or remains constant) when more predictors are added, even if they are not statistically significant.	Includes a penalty for the number of predictors. May decrease if added predictors do not improve the model sufficiently.
Interpretation	Proportion of the total variation in $\mathbf{Y}$ explained by the regression model.	Adjusted measure of explained variance, accounting for model complexity.
Range	Ranges between 0 and 1.	Can be lower than R^2, particularly when the model includes irrelevant predictors.
Usefulness	Useful for understanding the overall fit of the model.	Useful for comparing models with different numbers of predictors.

In multiple regression, R_a^2 provides a more reliable measure of model fit, especially when comparing models with different numbers of predictors.

In a regression model with coefficients $\beta = (\beta_0, \beta_1, \dots, \beta_{p-1})^\top$, the sums of squares are used to evaluate the contribution of predictors to explaining the variation in the response variable.

1. Model Sums of Squares:
   - SSM: Total model sum of squares, capturing the variation explained by all predictors:

$$SSM = SS(\beta_0, \beta_1, \dots, \beta_{p-1}).$$

2. Marginal Contribution:
   - SSM_m: Conditional model sum of squares, capturing the variation explained by predictors after accounting for others:

$$SSM_m = SS(\beta_0, \beta_1, \dots, \beta_{p-1} | \beta_0).$$

Decompositions of SSM_m

1. Sequential Sums of Squares (Type I SS)

- Definition:
  - Sequential SS depends on the order in which predictors are added to the model.
  - It represents the additional contribution of each predictor given only the predictors that precede it in the sequence.

- Formula:

$$SSM_m = SS(\beta_1|\beta_0) + SS(\beta_2|\beta_0, \beta_1) + \cdots + SS(\beta_{p-1}|\beta_0, \ldots, \beta_{p-2}).$$

- Key Points:
  - Sequential SS is not unique; it depends on the order of the predictors.
  - Default in many statistical software functions (e.g., `anova()` in R).

2. Marginal Sums of Squares (Type II SS)

- Definition:
  - Marginal SS evaluates the contribution of a predictor after accounting for all other predictors except those with which it is collinear.
  - It ignores hierarchical relationships or interactions, focusing on independent contributions.
- Formula: $SSM_m = SS(\beta_j|\beta_1, \ldots, \beta_{j-1}, \beta_{j+1}, \ldots, \beta_{p-1})$ where Type II SS evaluates the contribution of β_j while excluding any terms collinear with β_j.
- Key Points:
  - Type II SS is independent of predictor order.
  - Suitable for models without interaction terms or when predictors are balanced.

3. Partial Sums of Squares (Type III SS)

- Definition:
  - Partial SS evaluates the contribution of each predictor after accounting for all other predictors in the model.
  - It quantifies the unique contribution of a predictor, controlling for the presence of others.
- Formula:

$$SSM_m = SS(\beta_1|\beta_0, \beta_2, \ldots, \beta_{p-1}) + \cdots + SS(\beta_{p-1}|\beta_0, \beta_1, \ldots, \beta_{p-2}).$$

- Key Points:
  - Partial SS is unique for a given model.
  - More commonly used in practice for assessing individual predictor importance.

Comparison of Sequential, Marginal, and Partial SS

Aspect	Sequential SS (Type I)	Marginal SS (Type II)	Partial SS (Type III)
Dependency	Depends on the order in which predictors are entered.	Independent of order; adjusts for non-collinear predictors.	Independent of order; evaluates unique contributions.
Usage	Default in software functions like `anova()` (Type I SS).	Models without interactions or hierarchical dependencies.	Commonly used for hypothesis testing.
Interpretation	Measures the additional contribution of predictors in sequence.	Measures the contribution of a predictor, ignoring collinear terms.	Measures the unique contribution of each predictor.
Uniqueness	Not unique; changes with predictor order.	Unique for a given model without interactions.	Unique for a given model.

Practical Notes

- Use Type III SS (Partial SS) when:
  - The focus is on individual predictor contributions while accounting for all others.
  - Conducting hypothesis tests on predictors in complex models with interactions or hierarchical structures.
- Use Type II SS (Marginal SS) when:
  - Working with balanced datasets or models without interaction terms.
  - Ignoring interactions and focusing on independent effects.
- Use Type I SS (Sequential SS) when:
  - Interested in understanding the incremental contribution of predictors based on a specific order of entry (e.g., stepwise regression).

5.1.2.1 OLS Assumptions

- A1 Linearity
- A2 Full Rank
- A3 Exogeneity of Independent Variables
- A4 Homoskedasticity
- A5 Data Generation (Random Sampling)
- A6 Normal Distribution

5.1.2.1.1 A1 Linearity The linear regression model is expressed as:

$$A1 : y = \mathbf{x}\beta + \epsilon$$

This assumption is not restrictive since x can include nonlinear transformations (e.g., interactions, natural logarithms, quadratic terms).

However, when combined with A3 (Exogeneity of Independent Variables), linearity can become restrictive.

5.1.2.1.1.1 Log Model Variants Logarithmic transformations of variables allow for flexible modeling of nonlinear relationships. Common log model forms include:

Model	Form	Interpretation of β	In Words
Level-Level	$y = \beta_0 + \beta_1 x + \epsilon$	$\Delta y = \beta_1 \Delta x$	A unit change in x results in a β_1 unit change in y.
Log-Level	$\ln(y) = \beta_0 + \beta_1 x + \epsilon$	$\%\Delta y = 100\beta_1 \Delta x$	A unit change in x results in a $100\beta_1\%$ change in y.
Level-Log	$y = \beta_0 + \beta_1 \ln(x) + \epsilon$	$\Delta y = (\beta_1/100)\%\Delta x$	A 1% change in x results in a $(\beta_1/100)$ unit change in y.
Log-Log	$\ln(y) = \beta_0 + \beta_1 \ln(x) + \epsilon$	$\%\Delta y = \beta_1 \%\Delta x$	A 1% change in x results in a $\beta_1\%$ change in y.

5.1.2.1.1.2 Higher-Order Models Higher-order terms allow the effect of x_1 on y to depend on the level of x_1. For example:

$$y = \beta_0 + x_1\beta_1 + x_1^2\beta_2 + \epsilon$$

The partial derivative of y with respect to x_1 is:

$$\frac{\partial y}{\partial x_1} = \beta_1 + 2x_1\beta_2$$

- The effect of x_1 on y depends on the value of x_1.
- Partial Effect at the Average: $\beta_1 + 2E(x_1)\beta_2$.
- Average Partial Effect: $E(\beta_1 + 2x_1\beta_2)$.

5.1.2.1.1.3 Interaction Terms Interactions capture the joint effect of two variables. For example:

$$y = \beta_0 + x_1\beta_1 + x_2\beta_2 + x_1x_2\beta_3 + \epsilon$$

- β_1 is the average effect of a unit change in x_1 on y when $x_2 = 0$.
- The partial effect of x_1 on y, which depends on the level of x_2, is:

$$\beta_1 + x_2\beta_3.$$

5.1.2.1.2 A2 Full Rank The full rank assumption ensures the uniqueness and existence of the parameter estimates in the population regression equation. It is expressed as:

$$A2 : \text{rank}(E(\mathbf{x}'\mathbf{x})) = k$$

This assumption is also known as the identification condition.

Key Points

1. No Perfect Multicollinearity:
   - The columns of $\mathbf{x}$ (the matrix of predictors) must be linearly independent.
   - No column in $\mathbf{x}$ can be written as a linear combination of other columns.
2. Implications:
   - Ensures that each parameter in the regression equation is identifiable and unique.
   - Prevents computational issues, such as the inability to invert $\mathbf{x}'\mathbf{x}$, which is required for estimating $\hat{\beta}$.

Example of Violation

If two predictors, x_1 and x_2, are perfectly correlated (e.g., $x_2 = 2x_1$), the rank of $\mathbf{x}$ is reduced, and $\mathbf{x}'\mathbf{x}$ becomes singular. In such cases:

- The regression coefficients cannot be uniquely estimated.
- The model fails to satisfy the full rank assumption.

5.1.2.1.3 A3 Exogeneity of Independent Variables The exogeneity assumption ensures that the independent variables ($\mathbf{x}$) are not systematically related to the error term (ϵ). It is expressed as:

$$A3 : E[\epsilon|x_1, x_2, \dots, x_k] = E[\epsilon|\mathbf{x}] = 0$$

This assumption is often referred to as strict exogeneity or mean independence (see Correlation and Independence.

Key Points

1. Strict Exogeneity:
   - Independent variables carry no information about the error term ϵ.
   - By the [Law of Iterated Expectations], $E(\epsilon) = 0$, which can be satisfied by always including an intercept in the regression model.
2. Implication:
   - A3 implies:
   $$E(y|\mathbf{x}) = \mathbf{x}\beta,$$
   meaning the conditional mean function is a linear function of $\mathbf{x}$. This aligns with A1 Linearity.
3. Relationship with Independence:
   - Also referred to as mean independence, which is a weaker condition than full independence (see Correlation and Independence).

5.1.2.1.3.1 A3a: Weaker Exogeneity Assumption A weaker version of the exogeneity assumption is:

$$A3a : E(\mathbf{x_i'}\epsilon_i) = 0$$

This implies:

- The independent variables ($\mathbf{x}_i$) are uncorrelated with the error term (ϵ_i).
- Weaker than mean independence in A3.

Comparison Between A3 and A3a

Aspect	A3 (Strict Exogeneity)	A3a (Weaker Exogeneity)
Definition	$E(\epsilon\|\mathbf{x}) = 0$.	$E(\mathbf{x}_i'\epsilon_i) = 0$.
Strength	Stronger assumption; implies A3a.	Weaker assumption; does not imply A3.
Interpretation	Predictors provide no information about ϵ.	Predictors are uncorrelated with ϵ.
Causality	Enables causal interpretation.	Does not allow causal interpretations.

Notes on Practical Relevance

- Checking for Exogeneity:
  - Strict exogeneity cannot be tested directly, but violations can manifest as omitted variable bias, endogeneity, or measurement error.
  - Including all relevant predictors and ensuring accurate measurement can help satisfy this assumption.
- Violations of Exogeneity:
  - If A3 is violated, standard OLS estimates are biased and inconsistent.
  - In such cases, instrumental variable (IV) methods or other approaches may be required to correct for endogeneity.

5.1.2.1.4 A4 Homoskedasticity The homoskedasticity assumption ensures that the variance of the error term (ϵ) is constant across all levels of the independent variables ($\mathbf{x}$). It is expressed as:

$$A4 : \text{Var}(\epsilon|\mathbf{x}) = \text{Var}(\epsilon) = \sigma^2$$

Key Points

1. Definition:
   - The variance of the disturbance term ϵ is the same for all observations, regardless of the values of the predictors $\mathbf{x}$.
2. Practical Implication:
   - Homoskedasticity ensures that the errors do not systematically vary with the predictors.
   - This is critical for valid inference, as the standard errors of the coefficients rely on this assumption.
3. Violation (Heteroskedasticity):
   - If the variance of ϵ depends on $\mathbf{x}$, the assumption is violated.
   - Common signs include funnel-shaped patterns in residual plots or varying error sizes.

5.1.2.1.5 A5 Data Generation (Random Sampling) The random sampling assumption ensures that the observations $(y_i, x_{i1}, \dots, x_{ik-1})$ are drawn independently and identically distributed (iid) from the joint distribution of $(y, \mathbf{x})$. It is expressed as:

$$A5 : \{y_i, x_{i1}, \dots, x_{ik-1} : i = 1, \dots, n\}$$

Key Points

1. Random Sampling:
   - The dataset is assumed to be a random sample from the population.
   - Each observation is independent of others and follows the same probability distribution.
2. Implications:
   - With A3 (Exogeneity of Independent Variables) and A4 (Homoskedasticity), random sampling implies:
     - Strict Exogeneity:

       $$E(\epsilon_i | x_1, \dots, x_n) = 0$$

       Independent variables do not contain information for predicting ϵ.
     - Non-Autocorrelation:

       $$E(\epsilon_i \epsilon_j | x_1, \dots, x_n) = 0 \quad \text{for } i \neq j$$

       The error terms are uncorrelated across observations, conditional on the independent variables.
     - Variance of Errors:

       $$\text{Var}(\epsilon | \mathbf{X}) = \text{Var}(\epsilon) = \sigma^2 \mathbf{I}_n$$
3. When A5 May Not Hold:
   - In time series data, where observations are often autocorrelated.
   - In spatial data, where neighboring observations may not be independent.

Practical Considerations

- Time Series Data:
  - Use methods such as autoregressive models or generalized least squares (GLS) to address dependency in observations.
- Spatial Data:
  - Spatial econometric models may be required to handle correlation across geographic locations.
- Checking Random Sampling:
  - While true randomness cannot always be verified, exploratory analysis of the residuals (e.g., for patterns or autocorrelation) can help detect violations.

5.1.2.1.5.1 A5a: Stationarity in Stochastic Processes A stochastic process $\{x_t\}_{t=1}^T$ is stationary if, for every collection of time indices $\{t_1, t_2, \dots, t_m\}$, the joint distribution of:

$$x_{t_1}, x_{t_2}, \dots, x_{t_m}$$

is the same as the joint distribution of:

$$x_{t_1+h}, x_{t_2+h}, \dots, x_{t_m+h}$$

for any $h \geq 1$.

Key Points on Stationarity

1. Definition:
   - A stationary process has statistical properties (mean, variance, and covariance) that are invariant over time.
   - For example, the joint distribution for the first ten observations is identical to the joint distribution for the next ten observations, regardless of their position in time.
2. Implication:
   - Stationarity ensures that the relationships observed in the data remain consistent over time.
   - Independent draws automatically satisfy the stationarity condition.

Weak Stationarity

A stochastic process $\{x_t\}_{t=1}^T$ is weakly stationary if:

- The covariance between x_t and x_{t+h} depends only on the lag h and not on t.
- As $h \to \infty$, the covariance diminishes, meaning x_t and x_{t+h} become "almost independent."

Differences Between Stationarity and Weak Stationarity

Aspect	Stationarity	Weak Stationarity
Joint Distribution	Entire joint distribution remains unchanged over time.	Focuses only on the first two moments: mean and covariance.
Dependence Over Time	Observations at all lags are equally distributed.	Observations far apart are "almost independent."
Application	Ensures strong consistency in time-series processes.	More practical for many time-series applications.

Weak stationarity is often sufficient for many time-series analyses, especially when focusing on correlations and trends rather than full distributions.

Common Weakly Dependent Processes

1. Moving Average Process of Order 1 (MA(1))

An MA(1) process models the dependent variable y_t as a function of the current and one-period lagged stochastic error term:

$$y_t = u_t + \alpha_1 u_{t-1},$$

where u_t is white noise, independently and identically distributed (iid) with variance σ^2.

Key Properties

1. Mean:
$$E(y_t) = E(u_t) + \alpha_1 E(u_{t-1}) = 0$$
2. Variance:
$$\begin{aligned}\text{Var}(y_t) &= \text{Var}(u_t) + \alpha_1^2 \text{Var}(u_{t-1}) \\ &= \sigma^2 + \alpha_1^2 \sigma^2 \\ &= \sigma^2(1 + \alpha_1^2)\end{aligned}$$
   An increase in the absolute value of α_1 increases the variance.
3. Autocovariance:
   - For lag 1:
   $$\text{Cov}(y_t, y_{t-1}) = \alpha_1 \text{Var}(u_{t-1}) = \alpha_1 \sigma^2.$$
   - For lag 2 or greater:
   $$\text{Cov}(y_t, y_{t-2}) = 0.$$

The MA(1) process is invertible if $|\alpha_1| < 1$, allowing it to be rewritten as an autoregressive (AR) representation:

$$u_t = y_t - \alpha_1 u_{t-1}.$$

Invertibility implies that we can express the current observation in terms of past observations.

An MA(q) process generalizes the MA(1) process to include q lags:

$$y_t = u_t + \alpha_1 u_{t-1} + \cdots + \alpha_q u_{t-q},$$

where $u_t \sim WN(0, \sigma^2)$ (white noise with mean 0 and variance σ^2).

Key Characteristics:

1. Covariance Stationary:

- An MA(q) process is covariance stationary irrespective of the parameter values.

2. Invertibility:
   - An MA(q) process is invertible if the parameters satisfy certain conditions (e.g., $|\alpha_i| < 1$).
3. Autocorrelations:
   - The autocorrelations are nonzero for lags up to q but are 0 for lags beyond q.
4. Conditional Mean:
   - The conditional mean of y_t depends on the q lags, indicating "long-term memory."

Example: Autocovariance of an MA(1)

1. Lag 1:
$$\begin{aligned}\text{Cov}(y_t, y_{t-1}) &= \text{Cov}(u_t + \alpha_1 u_{t-1}, u_{t-1} + \alpha_1 u_{t-2}) \\ &= \alpha_1 \text{Var}(u_{t-1}) \\ &= \alpha_1 \sigma^2.\end{aligned}$$

2. Lag 2:
$$\begin{aligned}\text{Cov}(y_t, y_{t-2}) &= \text{Cov}(u_t + \alpha_1 u_{t-1}, u_{t-2} + \alpha_1 u_{t-3}) \\ &= 0.\end{aligned}$$

An MA process captures a linear relationship between the dependent variable y_t and the current and past values of a stochastic error term u_t. Its properties make it useful for modeling time-series data with limited memory and short-term dependencies.

2. Auto-Regressive Process of Order 1 (AR(1))

An auto-regressive process of order 1 (AR(1)) is defined as:

$$y_t = \rho y_{t-1} + u_t, \quad |\rho| < 1$$

where u_t represents independent and identically distributed (i.i.d.) random noise over t with variance σ^2.

Covariance at lag 1:

$$\begin{aligned}Cov(y_t, y_{t-1}) &= Cov(\rho y_{t-1} + u_t, y_{t-1}) \\ &= \rho Var(y_{t-1}) \\ &= \rho \frac{\sigma^2}{1-\rho^2}.\end{aligned}$$

Covariance at lag h:

$$Cov(y_t, y_{t-h}) = \rho^h \frac{\sigma^2}{1-\rho^2}.$$

Stationarity implies that the distribution of y_t does not change over time, requiring constant mean and variance. For this process:

1. Mean Stationarity:
   Assuming $E(y_t) = 0$, we have:

   $$y_t = \rho^t y_0 + \rho^{t-1} u_1 + \rho^{t-2} u_2 + \cdots + \rho u_{t-1} + u_t.$$

   If the initial observation $y_0 = 0$, then y_t is simply a weighted sum of the random shocks u_t from all prior time periods. Thus, $E(y_t) = 0$ for all t.

2. Variance Stationarity:
   The variance is computed as:

   $$Var(y_t) = Var(\rho y_{t-1} + u_t).$$

   Expanding and simplifying:

   $$Var(y_t) = \rho^2 Var(y_{t-1}) + Var(u_t) + 2\rho Cov(y_{t-1}, u_t).$$

   Since u_t is independent of y_{t-1}, $Cov(y_{t-1}, u_t) = 0$, giving:

   $$Var(y_t) = \rho^2 Var(y_{t-1}) + \sigma^2.$$

   Solving recursively, we find:

   $$Var(y_t) = \frac{\sigma^2}{1-\rho^2}.$$

For the variance to remain constant over time, it is required that $|\rho| < 1$ and $p \notin \{1, -1\}$.

Key Insights on Stationarity

1. Stationarity Requirement:
   The condition $|\rho| < 1$ ensures stationarity, as this guarantees that both the mean and variance of the process are constant over time.

2. Weak Dependence:
   As $|\rho| < 1$, the dependency between observations diminishes with increasing lag h, as seen from the covariance:

   $$Cov(y_t, y_{t-h}) = \rho^h \frac{\sigma^2}{1-\rho^2}.$$

To estimate an AR(1) process, we utilize the Yule-Walker equations, which relate the autocovariances of the process to its parameters.

Starting with the AR(1) process:

$$y_t = \epsilon_t + \phi y_{t-1},$$

multiplying both sides by $y_{t-\tau}$ and taking expectations, we get:

$$y_t y_{t-\tau} = \epsilon_t y_{t-\tau} + \phi y_{t-1} y_{t-\tau}.$$

For $\tau \geq 1$, the autocovariance $\gamma(\tau)$ satisfies:

$$\gamma(\tau) = \phi\gamma(\tau - 1).$$

Dividing through by the variance $\gamma(0)$, we obtain the autocorrelation:

$$\rho_\tau = \phi^\tau.$$

Thus, the autocorrelations decay geometrically as ϕ^τ, where $|\phi| < 1$ ensures stationarity and decay over time.

Generalizing to AR(p)

An AR(p) process extends the AR(1) structure to include p lags:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t.$$

Here, ϵ_t is white noise with $E(\epsilon_t) = 0$ and $Var(\epsilon_t) = \sigma^2$.

The AR(p) process is covariance stationary if the roots of the characteristic equation lie outside the unit circle:

$$1 - \phi_1 z - \phi_2 z^2 - \cdots - \phi_p z^p = 0.$$

For the AR(p) process, the autocorrelations ρ_τ decay more complexly compared to the AR(1). However, they still diminish over time, ensuring weak dependence among distant observations.

The Yule-Walker equations for an AR(p) process provide a system of linear equations to estimate the parameters $\phi_1, \phi_2, \ldots, \phi_p$:

$$\gamma(\tau) = \phi_1\gamma(\tau - 1) + \phi_2\gamma(\tau - 2) + \cdots + \phi_p\gamma(\tau - p), \quad \tau \geq 1.$$

This system can be written in matrix form for $\tau = 1, \ldots, p$ as:

$$\begin{bmatrix} \gamma(1) \\ \gamma(2) \\ \vdots \\ \gamma(p) \end{bmatrix} = \begin{bmatrix} \gamma(0) & \gamma(1) & \dots & \gamma(p-1) \\ \gamma(1) & \gamma(0) & \dots & \gamma(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ \gamma(p-1) & \gamma(p-2) & \dots & \gamma(0) \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_p \end{bmatrix}.$$

This system is solved to estimate the coefficients $\phi_1, \phi_2, \dots, \phi_p$.

---

3. ARMA(p, q) Process

An ARMA(p, q) process combines autoregressive (AR) and moving average (MA) components to model time series data effectively. The general form is:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t + \alpha_1 \epsilon_{t-1} + \alpha_2 \epsilon_{t-2} + \cdots + \alpha_q \epsilon_{t-q}.$$

A simple case of ARMA(1, 1) is given by:

$$y_t = \phi y_{t-1} + \epsilon_t + \alpha \epsilon_{t-1},$$

where:

- ϕ captures the autoregressive behavior,
- α controls the moving average component,
- ϵ_t represents white noise.

ARMA processes can capture seasonality and more complex dependencies than pure AR or MA models.

---

4. **Random Walk Process**

A **random walk** is a non-stationary process defined as:

$$y_t = y_0 + \sum_{s=1}^{t} u_s,$$

where u_s are i.i.d. random variables.

Properties:

1. **Non-Stationarity**:
   - If $y_0 = 0$, then $E(y_t) = 0$, but the variance increases over time:

$$Var(y_t) = t\sigma^2.$$

2. **Not Weakly Dependent**:

- The covariance of the process does not diminish with increasing lag h:

$$Cov\left(\sum_{s=1}^{t} u_s, \sum_{s=1}^{t-h} u_s\right) = (t-h)\sigma^2.$$

As h increases, the covariance remains large, violating the condition for weak dependence.

5.1.2.1.5.2 A5a: Stationarity and Weak Dependence in Time Series For time series data, the set $\{y_t, x_{t1}, \dots, x_{tk-1}\}$, where $t = 1, \dots, T$, must satisfy the conditions of **stationarity** and **weak dependence**. These properties are essential to ensure the consistency and efficiency of estimators in time-series regression models.

1. **Stationarity**: A stationary process has statistical properties (e.g., mean, variance, autocovariance) that remain constant over time. This ensures that the relationships in the data do not change as t progresses, making it possible to draw meaningful inferences.
2. **Weak Dependence**: Weak dependence implies that observations far apart in time are "almost independent." While there may be short-term correlations, these diminish as the time lag increases. This property ensures that the sample averages are representative of the population mean.

The Weak Law of Large Numbers provides a foundation for the consistency of sample means. If $\{z_t\}$ is a weakly dependent, stationary process with $E(|z_t|) < \infty$ and $E(z_t) = \mu$, then:

$$\frac{1}{T}\sum_{t=1}^{T} z_t \xrightarrow{p} \mu.$$

Interpretation:

- As the sample size $T \to \infty$, the sample mean $\bar{z} = \frac{1}{T}\sum_{t=1}^{T} z_t$ converges in probability to the true mean μ.
- This ensures the **consistency** of estimators based on time-series data.

The Central Limit Theorem extends the WLLN by describing the distribution of the sample mean. Under additional regularity conditions (e.g., finite variance) (Greene, 1990), the sample mean $\bar{z}$ satisfies:

$$\sqrt{T}(\bar{z} - \mu) \xrightarrow{d} N(0, B),$$

where:

$$B = \text{Var}(z_t) + 2\sum_{h=1}^{\infty} \text{Cov}(z_t, z_{t-h}).$$

Interpretation:

- The sample mean $\bar{z}$ is approximately normally distributed for large T.
- The variance of the limiting distribution, B, depends not only on the variance of z_t but also on the covariances between z_t and its past values.

5.1.2.1.6 A6 Normal Distribution A6: $\epsilon|\mathbf{x} \sim N(0, \sigma^2 I_n)$

The assumption here implies that the error term ϵ is normally distributed with mean zero and variance $\sigma^2 I_n$. This assumption is fundamental for statistical inference in linear regression models.

Using assumptions A1 Linearity, A2 Full Rank, and A3 Exogeneity of Independent Variables, we derive the identification (or orthogonality condition) for the population parameter β:

$$\begin{aligned}
y &= x\beta + \epsilon && \text{(A1: Model Specification)} \\
x'y &= x'x\beta + x'\epsilon && \text{(Multiply both sides by } x'\text{)} \\
E(x'y) &= E(x'x)\beta + E(x'\epsilon) && \text{(Taking expectation)} \\
E(x'y) &= E(x'x)\beta && \text{(A3: Exogeneity, } E(x'\epsilon) = 0\text{)} \\
[E(x'x)]^{-1}E(x'y) &= [E(x'x)]^{-1}E(x'x)\beta && \text{(Invertibility of } E(x'x)\text{, A2)} \\
[E(x'x)]^{-1}E(x'y) &= \beta && \text{(Simplified solution for } \beta\text{)}
\end{aligned}$$

Thus, β is identified as the vector of parameters that minimizes the expected squared error.

To find β, we minimize the expected value of the squared error:

$$\underset{\gamma}{\text{argmin}}\ E\big((y - x\gamma)^2\big)$$

The first-order condition is derived by taking the derivative of the objective function with respect to γ and setting it to zero:

$$\begin{aligned}
\frac{\partial E\big((y - x\gamma)^2\big)}{\partial \gamma} &= 0 \\
-2E(x'(y - x\gamma)) &= 0 \\
E(x'y) - E(x'x\gamma) &= 0 \\
E(x'y) &= E(x'x)\gamma \\
(E(x'x))^{-1}E(x'y) &= \gamma
\end{aligned}$$

This confirms that $\gamma = \beta$.

The second-order condition ensures that the solution minimizes the objective function. Taking the second derivative:

$$\frac{\partial^2 E((y - x\gamma)^2)}{\partial \gamma'^2} = 0 = 2E(x'x)$$

If assumption A3 Exogeneity of Independent Variables holds, $E(x'x)$ is positive semi-definite (PSD). Thus, $2E(x'x)$ is also PSD, ensuring a minimum.

5.1.2.1.7 Hierarchy of OLS Assumptions This table summarizes the hierarchical nature of assumptions required to derive different properties of the OLS estimator.

Table 5.16: Usage of Assumptions

Assumption	**Identification**		**Gauss-Markov (BLUE)**	**Classical LM (BUE)**
	Data Description	**Unbiasedness Consistency**	**Asymptotic Inference (z and Chi-squared)**	**Small-sample Inference (t and F)**
A2 Full Rank Variation in **X**	✓	✓	✓	✓
A5 Data Generation (Random Sampling) Random Sampling		✓	✓	✓
A1 Linearity Linearity in Parameters		✓	✓	✓
A3 Exogeneity of Independent Variables Zero Conditional Mean		✓	✓	✓
A4 Homoskedasticity **H** homoskedasticity			✓	✓

Assumption	**Identification Data Description**	**Unbiasedness Consistency**	**Gauss-Markov (BLUE) Asymptotic Inference (z and Chi-squared)**	**Classical LM (BUE) Small-sample Inference (t and F)**
A6 Normal Distribution Normality of Errors				✓

- **Identification Data Description:** Ensures the model is identifiable and coefficients can be estimated.
- **Unbiasedness Consistency:** Guarantees that OLS estimates are unbiased and converge to the true parameter values as the sample size increases.
- **Gauss-Markov (BLUE) and Asymptotic Inference:** Requires additional assumptions (e.g., homoskedasticity) to ensure minimum variance of estimators and valid inference using large-sample tests (z and chi-squared).
- **Classical LM (BUE) Small-sample Inference:** Builds on all previous assumptions and adds normality of errors for valid t and F tests in finite samples.

5.1.2.2 Theorems

5.1.2.2.1 Frisch–Waugh–Lovell Theorem {#Frisch–Waugh–Lovell Theorem} The Frisch–Waugh–Lovell (FWL) Theorem is a fundamental result in linear regression that allows for a deeper understanding of how coefficients are computed in a multiple regression setting (Lovell, 2008). Informally, it states:

> When estimating the effect of a subset of variables (X_1) on y in the presence of other variables (X_2), you can "partial out" the influence of X_2 from both y and X_1. Then, regressing the residuals of y on the residuals of X_1 produces coefficients for X_1 that are identical to those obtained from the full multiple regression.

Consider the multiple linear regression model:

$$\mathbf{y} = \mathbf{X} \; + \; = \mathbf{X_1} \; _1 + \mathbf{X_2} \; _2 +$$

where:

- y is an $n \times 1$ vector of the dependent variable.
- X_1 is an $n \times k_1$ matrix of regressors of interest.

- X_2 is an $n \times k_2$ matrix of additional regressors.
- β_1 and β_2 are coefficient vectors of sizes $k_1 \times 1$ and $k_2 \times 1$, respectively.
- ϵ is an $n \times 1$ error term vector.

This can be equivalently represented in partitioned matrix form as:

$$\begin{pmatrix} X_1'X_1 & X_1'X_2 \\ X_2'X_1 & X_2'X_2 \end{pmatrix} \begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \end{pmatrix} = \begin{pmatrix} X_1'y \\ X_2'y \end{pmatrix}$$

The OLS estimator for the vector $\begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$ is:

$$\begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \end{pmatrix} = \begin{pmatrix} X_1'X_1 & X_1'X_2 \\ X_2'X_1 & X_2'X_2 \end{pmatrix}^{-1} \begin{pmatrix} X_1'y \\ X_2'y \end{pmatrix}.$$

If we only want the coefficients on X_1, a known result from partitioned-inversion gives:

$$\hat{\beta}_1 = \left(X_1' M_2\, X_1\right)^{-1} X_1' M_2\, y,$$

where

$$M_2 = I - X_2 \left(X_2' X_2\right)^{-1} X_2'.$$

The matrix M_2 is often called the **residual-maker** or **annihilator** matrix for X_2. It is an $n \times n$ symmetric, idempotent projection matrix that projects any vector in $\mathbb{R}^n$ onto the orthogonal complement of the column space of X_2. M_2 satisfies $M_2^2 = M_2$, and $M_2 = M_2'$.

Intuitively, M_2 captures the part of y (and any other vector) that is orthogonal to the columns of X_2. This "partialling out" of X_2 from both y and X_1 lets us isolate $\hat{\beta}_1$.

Equivalently, we can also represent $\hat{\beta}_1$ as:

$$\hat{}_{\mathbf{1}} = (\mathbf{X_1'X_1})^{-\mathbf{1}}\mathbf{X_1'y} - (\mathbf{X_1'X_1})^{-\mathbf{1}}\mathbf{X_1'X_2}\hat{}_{\mathbf{2}}$$

From this equation, we can see that

1. **Betas from Multiple vs. Simple Regressions:**
   - The coefficients (β) from a multiple regression are generally **not the same** as the coefficients from separate individual simple regressions.
2. **Impact of Additional Variables (X_2):**
   - The inclusion of different sets of explanatory variables (X_2) affects all coefficient estimates, even for those in X_1.

3. **Special Cases:**
   - If $X_1'X_2 = 0$ (orthogonality between X_1 and X_2) or $\hat{\beta}_2 = 0$, the above points (1 and 2) do not hold. In such cases, there is no interaction between the coefficients in X_1 and X_2, making the coefficients in X_1 unaffected by X_2.

Steps in FWL:

1. **Partial Out** X_2 from y: Regress y on X_2 to obtain residuals:

$$\tilde{y} = M_2 y.$$

2. **Partial Out** X_2 from X_1: For each column of X_1, regress it on X_2 to obtain residuals:

$$\tilde{X}_1 = M_2 X_1.$$

3. **Regression of Residuals:** Regress $\tilde{y}$ on $\tilde{X}_1$:

$$\tilde{y} = \tilde{X}_1 \beta_1 + \text{error}.$$

The coefficients β_1 obtained here are identical to those from the full model regression:

$$y = X_1 \beta_1 + X_2 \beta_2 + \epsilon.$$

Why It Matters

1. **Interpretation of Partial Effects:** The FWL Theorem provides a way to interpret β_1 as the effect of X_1 on y after removing any linear dependence on X_2.
2. **Computational Simplicity:** It allows the decomposition of a large regression problem into smaller, computationally simpler pieces.

```r
# Simulate data
set.seed(123)
n <- 100
X1 <- matrix(rnorm(n * 2), n, 2)  # Two regressors of interest
X2 <- matrix(rnorm(n * 2), n, 2)  # Two additional regressors
beta1 <- c(2,-1)  # Coefficients for X1
beta2 <- c(1, 0.5)  # Coefficients for X2undefined
u <- rnorm(n)  # Error term
y <- X1 %*% beta1 + X2 %*% beta2 + u  # Generate dependent variable

# Full regression
```

```r
full_model <- lm(y ~ X1 + X2)
summary(full_model)
#>
#> Call:
#> lm(formula = y ~ X1 + X2)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -2.47336 -0.58010  0.07461  0.68778  2.46552
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  0.11614    0.10000   1.161    0.248
#> X11          1.77575    0.10899  16.293  < 2e-16 ***
#> X12         -1.14151    0.10204 -11.187  < 2e-16 ***
#> X21          0.94954    0.10468   9.071 1.60e-14 ***
#> X22          0.47667    0.09506   5.014 2.47e-06 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.9794 on 95 degrees of freedom
#> Multiple R-squared:  0.8297, Adjusted R-squared:  0.8225
#> F-statistic: 115.7 on 4 and 95 DF,  p-value: < 2.2e-16

# Step 1: Partial out X2 from y
y_residual <- residuals(lm(y ~ X2))

# Step 2: Partial out X2 from X1
X1_residual <- residuals(lm(X1 ~ X2))

# Step 3: Regress residuals
fwl_model <- lm(y_residual ~ X1_residual - 1)
summary(fwl_model)
#>
#> Call:
#> lm(formula = y_residual ~ X1_residual - 1)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -2.47336 -0.58010  0.07461  0.68778  2.46552
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> X1_residual1   1.7758     0.1073   16.55   <2e-16 ***
#> X1_residual2  -1.1415     0.1005  -11.36   <2e-16 ***
```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.9643 on 98 degrees of freedom
#> Multiple R-squared:  0.8109, Adjusted R-squared:  0.807
#> F-statistic: 210.1 on 2 and 98 DF,  p-value: < 2.2e-16

# Comparison of coefficients
cat("Full model coefficients (X1):", coef(full_model)[2:3], "\n")
#> Full model coefficients (X1): 1.775754 -1.141514
cat("FWL model coefficients:", coef(fwl_model), "\n")
#> FWL model coefficients: 1.775754 -1.141514
```

5.1.2.2.2 Gauss-Markov Theorem For a linear regression model:

$$\mathbf{y} = \mathbf{X} \ + \ ,$$

under the assumptions:

1. **A1**: Linearity of the model.
2. **A2**: Full rank of $\mathbf{X}$.
3. **A3**: Exogeneity of $\mathbf{X}$.
4. **A4**: Homoskedasticity of ϵ.

The Ordinary Least Squares estimator:

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-\mathbf{1}}\mathbf{X}'\mathbf{y},$$

is the **Best Linear Unbiased Estimator (BLUE)**. This means that $\hat{\beta}$ has the minimum variance among all linear unbiased estimators of β.

1. Unbiasedness

Suppose we consider **any** linear estimator of β of the form:

$$\tilde{\beta} = \mathbf{C}\,\mathbf{y},$$

where

- $\mathbf{y}$ is the $n \times 1$ vector of observations,
- $\mathbf{C}$ is a $k \times n$ matrix (with k the dimension of β) that depends **only** on the design matrix $\mathbf{X}$.

Our regression model is

$$\mathbf{y} = \mathbf{X}\beta + \epsilon, \quad E[\epsilon \mid \mathbf{X}] = \mathbf{0}, \quad \mathrm{Var}(\epsilon \mid \mathbf{X}) = \sigma^2\mathbf{I}.$$

We say $\tilde{\beta}$ is **unbiased** if its conditional expectation (given $\mathbf{X}$) equals the true parameter β:

$$E(\tilde{\beta} \mid \mathbf{X}) = E(\mathbf{C}\,\mathbf{y} \mid \mathbf{X}) = \beta.$$

1. Substitute $\mathbf{y} = \mathbf{X}\beta + \epsilon$:

$$E(\mathbf{C}\,\mathbf{y} \mid \mathbf{X}) = E\big(\mathbf{C}(\mathbf{X}\beta + \epsilon) \mid \mathbf{X}\big) = \mathbf{C}\,\mathbf{X}\,\beta + \mathbf{C}\,E(\epsilon \mid \mathbf{X}) = \mathbf{C}\,\mathbf{X}\,\beta.$$

2. For this to hold for *all* β, we require

$$\mathbf{C}\,\mathbf{X} = \mathbf{I}.$$

In other words, $\mathbf{C}$ must be a "right-inverse" of $\mathbf{X}$.

On the other hand, the OLS estimator $\hat{\beta}$ is given by

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\,\mathbf{y}.$$

You can verify:

- Let $\mathbf{C}_{\text{OLS}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$.
- Then
$$\mathbf{C}_{\text{OLS}}\,\mathbf{X} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\,\mathbf{X} = \mathbf{I}.$$
- By the argument above, this makes $\hat{\beta}$ unbiased.

Hence, any linear estimator $\tilde{\beta} = \mathbf{C}\,\mathbf{y}$ that is unbiased must satisfy $\mathbf{C}\,\mathbf{X} = \mathbf{I}$.

2. Minimum Variance (Gauss–Markov Part)

Among all estimators of the form $\tilde{\beta} = \mathbf{C}\,\mathbf{y}$ that are unbiased (so $\mathbf{C}\,\mathbf{X} = \mathbf{I}$), **OLS** achieves the smallest covariance matrix.

- Variance of a General Unbiased Estimator

If $\tilde{\beta} = \mathbf{C}\,\mathbf{y}$ with $\mathbf{C}\,\mathbf{X} = \mathbf{I}$, then:

$$\text{Var}(\tilde{\beta} \mid \mathbf{X}) = \text{Var}(\mathbf{C}\,\mathbf{y} \mid \mathbf{X}) = \mathbf{C}\,\text{Var}(\mathbf{y} \mid \mathbf{X})\,\mathbf{C}' = \mathbf{C}(\sigma^2\mathbf{I})\mathbf{C}' = \sigma^2\,\mathbf{C}\,\mathbf{C}'.$$

- Variance of the OLS Estimator

For OLS, $\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\,\mathbf{y}$. Thus,

$$\text{Var}(\hat{\beta} \mid \mathbf{X}) = \sigma^2\,(\mathbf{X}'\mathbf{X})^{-1}.$$

- Comparing $\text{Var}(\tilde{\beta})$ to $\text{Var}(\hat{\beta})$

We want to show:

$$\mathrm{Var}(\tilde{\beta} \mid \mathbf{X}) - \mathrm{Var}(\hat{\beta} \mid \mathbf{X}) \text{ is positive semi-definite.}$$

1. Since both $\tilde{\beta}$ and $\hat{\beta}$ are unbiased, we know:

$$\mathbf{C}\,\mathbf{X} = \mathbf{I}, \quad (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\,\mathbf{X} = \mathbf{I}.$$

2. One can show algebraically (as in the proof provided above) that

$$\mathrm{Var}(\tilde{\beta} \mid \mathbf{X}) - \mathrm{Var}(\hat{\beta} \mid \mathbf{X}) = \sigma^2\big[\mathbf{C}\mathbf{C}' - (\mathbf{X}'\mathbf{X})^{-1}\big].$$

   Under the condition $\mathbf{C}\,\mathbf{X} = \mathbf{I}$, the difference $\mathbf{C}\mathbf{C}' - (\mathbf{X}'\mathbf{X})^{-1}$ is positive semi-definite.

3. Positive semi-definite difference means

$$\mathbf{v}'\Big(\mathbf{C}\mathbf{C}' - (\mathbf{X}'\mathbf{X})^{-1}\Big)\mathbf{v} \geq 0 \quad \text{for all vectors } \mathbf{v}.$$

Hence, $\hat{\beta}$ has the smallest variance (in the sense of covariance matrices) among all linear unbiased estimators $\tilde{\beta} = \mathbf{C}\,\mathbf{y}$.

Summary of the Key Points

1. **Unbiasedness:**
   A linear estimator $\tilde{\beta} = \mathbf{C}\,\mathbf{y}$ is unbiased if $E(\tilde{\beta} \mid \mathbf{X}) = \beta$.
   This forces $\mathbf{C}\,\mathbf{X} = \mathbf{I}$.

2. **OLS is Unbiased:**
   The OLS estimator $\hat{\beta} = (X'X)^{-1}X'\,y$ satisfies $(X'X)^{-1}X'\,X = I$, hence is unbiased.

3. **OLS has Minimum Variance:**
   Among all $\mathbf{C}$ that satisfy $\mathbf{C}\,\mathbf{X} = \mathbf{I}$, the matrix $(\mathbf{X}'\mathbf{X})^{-1}$ gives the smallest possible $\mathrm{Var}(\tilde{\beta})$.
   In matrix form, $\mathrm{Var}(\tilde{\beta}) - \mathrm{Var}(\hat{\beta})$ is positive semi-definite, showing OLS is optimal (the **Best Linear Unbiased Estimator**, BLUE).

5.1.2.3 Finite Sample Properties

The finite sample properties of an estimator are considered when the sample size n is fixed (not asymptotically large). Key properties include **bias**, **distribution**, and **standard deviation** of the estimator.

Bias measures how close an estimator is, on average, to the true parameter value β. It is defined as:

$$\text{Bias} = E(\hat{\beta}) - \beta$$

Where:

- β: True parameter value.
- $\hat{\beta}$: Estimator for β.

Unbiased Estimator: An estimator is unbiased if:

$$\text{Bias} = E(\hat{\beta}) - \beta = 0 \quad \text{or equivalently} \quad E(\hat{\beta}) = \beta$$

This means the estimator will produce estimates that are, on average, equal to the value it is trying to estimate.

An estimator is a function of random variables (data). Its **distribution** describes how the estimates vary across repeated samples. Key aspects include:

- **Center**: Mean of the distribution, which relates to bias.
- **Spread**: Variability of the estimator, captured by its standard deviation or variance.

The **standard deviation** of an estimator measures the spread of its sampling distribution. It indicates the variability of the estimator across different samples.

5.1.2.3.1 Ordinary Least Squares Properties Under the standard assumptions for OLS:

1. A1: The relationship between Y and X is linear.
2. A2: The matrix $\mathbf{X}'\mathbf{X}$ is invertible.
3. A3: $E(\epsilon|X) = 0$ (errors are uncorrelated with predictors).

OLS is unbiased under these assumptions. The proof is as follows:

$$
\begin{aligned}
E(\hat{\beta}) &= E((\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}) && \text{A2} \\
&= E((\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'(\mathbf{X}\ +\ )) && \text{A1} \\
&= E((\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{X}\ + (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\ ) \\
&= E(\beta + (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\ ) \\
&= \beta + E((\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\ ) \\
&= \beta + E(E((\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\ |X)) && \text{LIE (Law of Iterated Expectation)} \\
&= \beta + E((\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'E(\epsilon|X)) \\
&= \beta + E((\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' \cdot 0) && \text{A3} \\
&= \beta
\end{aligned}
$$

Key Points:

1. **Linearity of Expectation**: Used to separate terms involving β and ϵ.
2. Law of Iterated Expectation **(LIE)**: Simplifies nested expectations.
3. **Exogeneity of Errors (A3)**: Ensures $E(\epsilon|X) = 0$, eliminating bias.

Implications of Unbiasedness

- OLS estimators are centered around the true value β across repeated samples.
- In small samples, OLS estimators may exhibit variability, but their expected value remains β.

If the assumption of exogeneity (A3) is violated, the OLS estimator becomes **biased**. Specifically, omitted variables or endogeneity can introduce systematic errors into the estimation.

From the **Frisch-Waugh-Lovell Theorem**:

- If an omitted variable $\hat{\beta}_2 \neq 0$ (non-zero effect) and the omitted variable is correlated with the included predictors ($\mathbf{X_1'X_2} \neq 0$), then the OLS estimator will be biased.
- This bias arises because the omitted variable contributes to the variation in the dependent variable, but its effect is incorrectly attributed to other predictors.

5.1.2.3.2 Conditional Variance of OLS Estimator Under assumptions A1, A2, A3, and A4, the **conditional variance of the OLS estimator** is:

$$
\begin{aligned}
Var(\hat{\beta}|\mathbf{X}) &= Var(\beta + (\mathbf{X'X})^{-\mathbf{1}}\mathbf{X'}\ |\mathbf{X}) && \text{A1-A2} \\
&= Var((\mathbf{X'X})^{-\mathbf{1}}\mathbf{X'}\ |\mathbf{X}) \\
&= (\mathbf{X'X})^{-\mathbf{1}}\mathbf{X'}Var(\epsilon|\mathbf{X})\mathbf{X}(\mathbf{X'X})^{-\mathbf{1}} \\
&= (\mathbf{X'X})^{-\mathbf{1}}\mathbf{X'}\sigma^2 I\mathbf{X}(\mathbf{X'X})^{-\mathbf{1}} && \text{A4} \\
&= \sigma^2(\mathbf{X'X})^{-\mathbf{1}}
\end{aligned}
$$

This result shows that the variance of $\hat{\beta}$ depends on:

1. σ^2: The variance of the errors.
2. $\mathbf{X'X}$: The information content in the design matrix $\mathbf{X}$.

5.1.2.3.3 Sources of Variation in OLS Estimator

1. **Unexplained Variation in the Dependent Variable**: $\sigma^2 = Var(\epsilon_i|\mathbf{X})$

- Large σ^2 indicates that the amount of unexplained variation (noise) is high relative to the explained variation ($\mathbf{x_i}$).
- This increases the variance of the OLS estimator.

2. **Small Variation in Predictor Variables**

- If the variance of predictors $(Var(x_{i1}), Var(x_{i2}), ...)$ is small, the design matrix $\mathbf{X}$ lacks information, leading to:
  - High variability in $\hat{\beta}$.
  - Potential issues in estimating coefficients accurately.
- **Small sample size** exacerbates this issue, as fewer observations reduce the robustness of parameter estimates.

3. **Correlation Between Explanatory Variables (Collinearity)**

- **Strong correlation** among explanatory variables creates problems:
  - x_{i1} being highly correlated with a linear combination of $1, x_{i2}, x_{i3}, ...$ contributes to inflated standard errors for $\hat{\beta}_1$.
  - Including many irrelevant variables exacerbates this issue.
- **Perfect Collinearity**:
  - If x_1 is perfectly determined by a linear combination of other predictors, the matrix $\mathbf{X'X}$ becomes singular.
  - This violates A2, making OLS impossible to compute.
- **Multicollinearity**:
  - If x_1 is highly correlated (but not perfectly) with a linear combination of other variables, the variance of $\hat{\beta}_1$ increases.
  - Multicollinearity does not violate OLS assumptions but weakens inference by inflating standard errors.

5.1.2.3.4 Standard Errors Standard errors measure the variability of an estimator, specifically the standard deviation of $\hat{\beta}$. They are crucial for inference, as they quantify the uncertainty associated with parameter estimates.

The variance of the OLS estimator $\hat{\beta}$ is:

$$Var(\hat{\beta}|\mathbf{X}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-\mathbf{1}}$$

Where:

- σ^2: Variance of the error terms.
- $(\mathbf{X}'\mathbf{X})^{-\mathbf{1}}$: Inverse of the design matrix product, capturing the geometry of the predictors.

Estimation of σ^2

Under assumptions A1 through A5, we can estimate σ^2 as:

$$s^2 = \frac{1}{n-k}\sum_{i=1}^{n} e_i^2 = \frac{1}{n-k}SSR$$

Where:

- n: Number of observations.
- k: Number of predictors, including the intercept.
- e_i: Residuals from the regression model ($e_i = y_i - \hat{y}_i$).
- SSR: Sum of squared residuals ($\sum e_i^2$).

The degrees of freedom adjustment $(n-k)$ accounts for the fact that residuals e_i are not true errors ϵ_i. Since the regression model uses k parameters, we lose k degrees of freedom in estimating variance.

The standard error for σ is:

$$s = \sqrt{s^2}$$

However, s is a biased estimator of σ due to Jensen's Inequality.

The standard error of each regression coefficient $\hat{\beta}_{j-1}$ is:

$$SE(\hat{\beta}_{j-1}) = s\sqrt{[(\mathbf{X}'\mathbf{X})^{-1}]_{jj}}$$

Alternatively, it can be expressed in terms of SST_{j-1} and R^2_{j-1}:

$$SE(\hat{\beta}_{j-1}) = \frac{s}{\sqrt{SST_{j-1}(1 - R^2_{j-1})}}$$

Where:

- SST_{j-1}: Total sum of squares for x_{j-1} from a regression of x_{j-1} on all other predictors.
- R^2_{j-1}: Coefficient of determination for the same regression.

This formulation highlights the role of multicollinearity, as R^2_{j-1} reflects the correlation between x_{j-1} and other predictors.

5.1.2.3.5 Summary of Finite Sample Properties of OLS Under Different Assumptions

1. **Under A1-A3**: OLS is unbiased.

$$E(\hat{\beta}) = \beta$$

2. **Under A1-A4**: The variance of the OLS estimator is:

$$Var(\hat{\beta}|\mathbf{X}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-\mathbf{1}}$$

3. **Under A1-A4, A6**: The OLS estimator is normally distributed:

$$\hat{\beta} \sim N(\beta, \sigma^2(\mathbf{X}'\mathbf{X})^{-\mathbf{1}})$$

4. **Under A1-A4, Gauss-Markov Theorem**: OLS is BLUE (Best Linear Unbiased Estimator).

5. **Under A1-A5**: The standard errors for $\hat{\beta}$ are unbiased estimators of the standard deviation of $\hat{\beta}$.

5.1.2.4 Large Sample Properties

Large sample properties provide a framework to evaluate the quality of estimators when finite sample properties are either uninformative or computationally infeasible. This perspective becomes crucial in modern data analysis, especially for methods like GLS or MLE, where assumptions for finite sample analysis may not hold.

Table 5.17: When to Use Finite vs. Large Sample Analysis

Aspect	Finite Sample Properties	Large Sample Properties
Applicability	Limited to fixed sample sizes	Relevant for $n \to \infty$
Exactness	Exact results (e.g., distributions, unbiasedness)	Approximate results
Assumptions	May require stronger assumptions (e.g., normality, independence)	Relies on asymptotic approximations (e.g., CLT, LLN)
Estimator Behavior	Performance may vary significantly	Estimators stabilize and improve in accuracy
Ease of Use	Often complex due to reliance on exact distributions	Simplifies analysis by leveraging asymptotic approximations
Real-World Relevance	More realistic for small datasets	More relevant for large datasets

- **Finite Sample Analysis**:
  - Small sample sizes (e.g., $n < 30$).
  - Critical for studies where exact results are needed.
  - Useful in experimental designs and case studies.
- **Large Sample Analysis**:
  - Large datasets (e.g., $n > 100$).
  - Necessary when asymptotic approximations improve computational simplicity.

Key Concepts:

1. **Consistency**:
   - Consistency ensures that an estimator converges in probability to the true parameter value as the sample size increases.
   - Mathematically, an estimator $\widehat{\theta}$ is consistent for θ if:

$$\widehat{\theta}_n \to^p \theta \quad \text{as } n \to \infty.$$

   - Consistency does not imply unbiasedness, and unbiasedness does not guarantee consistency.
2. **Asymptotic Distribution**:
   - The limiting distribution describes the shape of the scaled estimator as $n \to \infty$.

- Asymptotic distributions often follow normality due to the Central Limit Theorem, which underpins much of inferential statistics.

3. **Asymptotic Variance**:
   - Represents the spread of the estimator with respect to its limiting distribution.
   - Smaller asymptotic variance implies greater precision in large samples.

Motivation

Finite Sample Properties, such as unbiasedness, rely on strong assumptions like:

- A1 Linearity
- A3 Exogeneity of Independent Variables
- A4 Homoskedasticity
- A6 Normal Distribution

When these assumptions are violated or impractical to verify, finite sample properties lose relevance. In such cases, Large Sample Propertiesserve as an essential alternative for evaluating estimators.

For example, let the conditional expectation function (CEF) be:

$$\mu(\mathbf{X}) = E(y|\mathbf{X}),$$

which represents the **minimum mean squared predictor** over all possible functions $f(\mathbf{X})$:

$$\min_f E((y - f(\mathbf{X}))^2).$$

Under the assumptions A1 and A3, the CEF simplifies to:

$$\mu(\mathbf{X}) = \mathbf{X}\beta.$$

The **linear projection** is given by:

$$L(y|1, \mathbf{X}) = \gamma_0 + \mathbf{X}\mathrm{Var}(\mathbf{X})^{-1}\mathrm{Cov}(\mathbf{X}, y),$$

where:

$$\gamma = \mathbf{X}\mathrm{Var}(\mathbf{X})^{-1}\mathrm{Cov}(\mathbf{X}, y).$$

This linear projection minimizes the mean squared error:

$$(\gamma_0, \gamma) = \arg\min_{(a,b)} E\left[(E(y|\mathbf{X}) - (a + \mathbf{X}b))^2\right].$$

Implications for OLS

- **Consistency**: OLS is always consistent for the linear projection, ensuring convergence to the true parameter value as $n \to \infty$.

- **Causal Interpretation**: The linear projection has no inherent causal interpretation—it approximates the conditional mean function.
- **Assumption Independence**: Unlike the CEF, the linear projection does not depend on assumptions A1 and A3.

Evaluating Estimators via Large Sample Properties

1. **Consistency**:
   - Measures the estimator's centrality to the true value.
   - A consistent estimator ensures that with larger samples, estimates become arbitrarily close to the population parameter.
2. **Limiting Distribution**:
   - Helps infer the sampling behavior of the estimator as n grows.
   - Often approximated by a normal distribution for practical use in hypothesis testing and confidence interval construction.
3. **Asymptotic Variance**:
   - Quantifies the dispersion of the estimator around its limiting distribution.
   - Smaller variance is desirable for greater reliability.

An estimator $\hat{\theta}$ is **consistent** for a parameter θ if, as the sample size n increases, $\hat{\theta}$ converges in probability to θ:

$$\hat{\theta}_n \to^p \theta \quad \text{as } n \to \infty.$$

1. **Convergence in Probability**:
   - The probability that $\hat{\theta}$ deviates from θ by more than a small margin (no matter how small) approaches zero as n increases.

   Formally:
   $$\forall \epsilon > 0, \quad P(|\hat{\theta}_n - \theta| > \epsilon) \to 0 \quad \text{as } n \to \infty.$$

2. **Interpretation**: Consistency ensures that the estimator becomes arbitrarily close to the true population parameter θ as the sample size grows.

3. **Asymptotic Behavior**: Large sample properties rely on consistency to provide valid approximations of an estimator's behavior in finite samples.

Relationship Between Consistency and Unbiasedness

- **Unbiasedness**:
  - An estimator $\hat{\theta}$ is unbiased if its expected value equals the true parameter:
    $$E(\hat{\theta}) = \theta.$$

  - Unbiasedness is a finite-sample property and does not depend on the sample size.
- **Consistency**:
  - Consistency is a large-sample property and requires $\hat{\theta}$ to converge to θ as $n \to \infty$.

Important Distinctions

1. **Unbiasedness Does Not Imply Consistency**:
   - Example: Consider an unbiased estimator with extremely high variance that does not diminish as n increases. Such an estimator does not converge to θ in probability.
2. **Consistency Does Not Imply Unbiasedness**:
   - Example: $\hat{\theta}_n = \frac{n-1}{n}\theta$ is biased for all finite n, but as $n \to \infty$, $\hat{\theta}_n \to^p \theta$, making it consistent.

From the OLS formula:

$$\hat{\beta} = (\mathbf{X'X})^{-\mathbf{1}}\mathbf{X'y},$$

we can expand as:

$$\hat{\beta} = (\sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{x_i'x_i})^{-\mathbf{1}} \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{x_i'y_i},$$

or equivalently:

$$\hat{\beta} = (n^{-1}\sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{x_i'x_i})^{-\mathbf{1}} n^{-1}\sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{x_i'y_i}.$$

Taking the probability limit under the assumptions A2 and A5, we apply the Weak Law of Large Numbers (for a random sample, averages converge to expectations as $n \to \infty$):

$$plim(\hat{\beta}) = plim((n^{-1}\sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{x_i'x_i})^{-\mathbf{1}}) plim(n^{-1}\sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{x_i'y_i}),$$

which simplifies to:

$$\begin{aligned} plim(\hat{\beta}) &= (E(\mathbf{x_i'x_i}))^{-1}E(\mathbf{x_i'y_i}) && \text{A1} \\ &= (E(\mathbf{x_i'x_i}))^{-1}(E(\mathbf{x_i'x_i}\,\beta) + E(\mathbf{x_i}\ \ \mathbf{i})) && (A3a) \\ &= (E(\mathbf{x_i'x_i}))^{-1}E(\mathbf{x_i'x_i})\,\beta && (A2) \\ &= \beta \end{aligned}$$

Proof:

For a model of $y_i = x_i\beta + \epsilon_i$, where β is the true parameter vector, ϵ_i is the random error:

Expanding $E(x_i y_i)$:

$$E(x_i' y_i) = E(x_i'(x_i\beta + \epsilon_i))$$

By the linearity of expectation:

$$E(x_i' y_i) = E(x_i' x_i \beta) + E(x_i \epsilon_i)$$

The second term $E(x_i \epsilon_i) = 0$ under assumption A3a.

Thus,

$$E(x_i' y_i) = E(x_i' x_i)\beta.$$

Consistency of OLS

Hence, in short, under the assumptions:

- A1 Linearity
- A2 Full Rank
- A3a: Weaker Exogeneity Assumption
- A5 Data Generation (Random Sampling)

the term $E(\mathbf{x_i'}_{\ \mathbf{i}}) = 0$, and the OLS estimator is **consistent**:

$$plim(\hat{\beta}) = \beta.$$

However, OLS consistency does not guarantee unbiasedness in small samples.

5.1.2.4.1 Asymptotic Distribution of OLS Under the same assumptions :

- A1 Linearity
- A2 Full Rank
- A3a: Weaker Exogeneity Assumption
- A5 Data Generation (Random Sampling)

and if $\mathbf{x_i'x_i}$ has finite first and second moments (required for the Central Limit Theorem), we have:

1. Convergence of $n^{-1}\sum_{i=1}^{n} \mathbf{x_i'x_i}$:

$$n^{-1}\sum_{i=1}^{n} \mathbf{x_i'x_i} \to^p E(\mathbf{x_i'x_i}).$$

2. Asymptotic normality of $\sqrt{n}(n^{-1}\sum_{i=1}^{n}\mathbf{x}_{\mathbf{i}}'\,_{\mathbf{i}})$:

$$\sqrt{n}(n^{-1}\sum_{i=1}^{n}\mathbf{x}_{\mathbf{i}}'\,_{\mathbf{i}}) \to^{d} N(0,\mathbf{B}),$$

   where $\mathbf{B} = Var(\mathbf{x}_{\mathbf{i}}'\,_{\mathbf{i}})$.

From these results, the scaled difference between $\hat{\beta}$ and β follows:

$$\sqrt{n}(\hat{\beta}-\beta) = (n^{-1}\sum_{i=1}^{n}\mathbf{x}_{\mathbf{i}}'\mathbf{x}_{\mathbf{i}})^{-1}\sqrt{n}(n^{-1}\sum_{i=1}^{n}\mathbf{x}_{\mathbf{i}}'\,_{\mathbf{i}}).$$

By the Central Limit Theorem:

$$\sqrt{n}(\hat{\beta}-\beta) \to^{d} N(0,\Sigma),$$

where:

$$\Sigma = (E(\mathbf{x}_{\mathbf{i}}'\mathbf{x}_{\mathbf{i}}))^{-1}\mathbf{B}(E(\mathbf{x}_{\mathbf{i}}'\mathbf{x}_{\mathbf{i}}))^{-1}.$$

The sandwich form is Σ is standard.

Implications for Homoskedasticity (A4) vs. Heteroskedasticity

1. No Homoskedasticity (A4 Homoskedasticity) needed:

   - the CLT and the large-sample distribution of $\hat{\beta}$ do *not* require homoskedasticity. The only place homoskedasticity would simplify things is that

$$\mathbf{B} = Var(\mathbf{x}_{\mathbf{i}}'\,_{\mathbf{i}}) = \sigma^2 E(\mathbf{x}_{\mathbf{i}}'\mathbf{x}_{\mathbf{i}}),$$

   only if $Var(\epsilon_i|\mathbf{x}_i)\sigma^2$

   Then

$$\Sigma = \sigma^2(E(\mathbf{x}_{\mathbf{i}}'\mathbf{x}_{\mathbf{i}}))^{-1}.$$

2. Adjusting for Heteroskedasticity:

   - In practice, σ_i^2 can vary with $\mathbf{x}_i$, leading to heteroskedasticity.
   - The standard OLS formula for $Var(\hat{\beta})$ is inconsistent under heteroskedasticity, so one uses *robust (White) standard errors.*
   - Heteroskedasticity can arise from (but not limited to):
     - Limited dependent variables.
     - Dependent variables with large or skewed ranges.

5.1.2.4.2 Derivation of Asymptotic Variance The asymptotic variance of the OLS estimator is derived as follows:

$$\Sigma = (E(\mathbf{x_i'x_i}))^{-1}\mathbf{B}(E(\mathbf{x_i'x_i}))^{-1}.$$

1. Substituting $\mathbf{B} = Var(\mathbf{x_i'}\epsilon_i)$:

$$\Sigma = (E(\mathbf{x_i'x_i}))^{-1}Var(\mathbf{x_i'}\epsilon_i)(E(\mathbf{x_i'x_i}))^{-1}.$$

2. Using the definition of variance:

$$\Sigma = (E(\mathbf{x_i'x_i}))^{-1}E[(\mathbf{x_i'}\epsilon_i - 0)(\mathbf{x_i'}\epsilon_i - 0)'](E(\mathbf{x_i'x_i}))^{-1}.$$

3. By the [Law of Iterated Expectations] and A3a: Weaker Exogeneity Assumption, we have:

$$E[(\mathbf{x_i'}\epsilon_i)(\mathbf{x_i'}\epsilon_i)'] = E[E(\epsilon_i^2|\mathbf{x_i})\mathbf{x_i'x_i}],$$

4. Assuming homoskedasticity (A4 Homoskedasticity), $E(\epsilon_i^2|\mathbf{x_i}) = \sigma^2$, so:

$$\Sigma = (E(\mathbf{x_i'x_i}))^{-1}\sigma^2 E(\mathbf{x_i'x_i})(E(\mathbf{x_i'x_i}))^{-1}.$$

5. Simplifying:

$$\Sigma = \sigma^2(E(\mathbf{x_i'x_i}))^{-1}.$$

Hence, under the assumptions:

- A1 Linearity
- A2 Full Rank
- A3a: Weaker Exogeneity Assumption
- A4 Homoskedasticity
- A5 Data Generation (Random Sampling)

we have

$$\sqrt{n}(\hat{\beta} - \beta) \to^d N(0, \sigma^2(E(\mathbf{x_i'x_i}))^{-1}).$$

The asymptotic variance provides an approximation of the scaled estimator's variance for large n. This leads to:

$$Avar(\sqrt{n}(\hat{\beta} - \beta)) = \sigma^2(E(\mathbf{x_i'x_i}))^{-1}.$$

The finite sample variance of an estimator can be approximated using the asymptotic variance for large sample sizes:

$$Avar(\sqrt{n}(\hat{\beta}-\beta)) \approx Var(\sqrt{n}(\hat{\beta}-\beta))$$
$$Avar(\sqrt{n}(\hat{\beta}-\beta))/n \approx Var(\sqrt{n}(\hat{\beta}-\beta))/n = Var(\hat{\beta})$$

However, it is critical to note that **asymptotic variance** ($Avar(.)$) does not behave in the same manner as finite sample variance ($Var(.)$). This distinction is evident in the following expressions:

$$\begin{aligned} Avar(\sqrt{n}(\hat{\beta}-\beta))/n &\neq Avar(\sqrt{n}(\hat{\beta}-\beta)/\sqrt{n}) \\ &\neq Avar(\hat{\beta}) \end{aligned}$$

Thus, while $Avar(.)$ provides a useful approximation for large samples, its conceptual properties differ from finite sample variance.

In Finite Sample Properties, the standard errors are calculated as estimates of the conditional standard deviation:

$$SE_{fs}(\hat{\beta}_{j-1}) = \sqrt{\hat{Var}(\hat{\beta}_{j-1}|\mathbf{X})} = \sqrt{s^2 \cdot [(\mathbf{X}'\mathbf{X})^{-1}]_{jj}},$$

where:

- s^2 is the estimator of the error variance,
- $[(\mathbf{X}'\mathbf{X})^{-1}]_{jj}$ represents the jth diagonal element of the inverse design matrix.

In contrast, in Large Sample Properties, the standard errors are calculated as estimates of the square root of the asymptotic variance:

$$SE_{ls}(\hat{\beta}_{j-1}) = \sqrt{\hat{Avar}(\sqrt{n}\hat{\beta}_{j-1})/n} = \sqrt{s^2 \cdot [(\mathbf{X}'\mathbf{X})^{-1}]_{jj}}.$$

Interestingly, the standard error estimator is **identical** for both finite and large samples:

- The expressions for SE_{fs} and SE_{ls} are mathematically the same.
- However, they are conceptually estimating two **different quantities**:
  - **Finite Sample Standard Error**: An estimate of the conditional standard deviation of $\hat{\beta}_{j-1}$ given $\mathbf{X}$.
  - **Large Sample Standard Error**: An estimate of the square root of the asymptotic variance of $\hat{\beta}_{j-1}$.

The assumptions required for these estimators to be valid differ in their stringency:

1. **Finite Sample Variance (Conditional Variance):**

- Requires stronger assumptions (A1-A5).

2. **Asymptotic Variance:**
   - Valid under weaker assumptions (A1, A2, A3a, A4, A5).

This distinction highlights the utility of asymptotic properties in providing robust approximations when finite sample assumptions may not hold.

5.1.2.5 Diagnostics

5.1.2.5.1 Normality of Errors Ensuring the normality of errors is a critical assumption in many regression models. Deviations from this assumption can impact inference and model interpretation. For diagnoses assessing normality, see Normality Assessment.

Plots are invaluable for visual inspection of normality. One common approach is the Q-Q plot, which compares the quantiles of the residuals against those of a standard normal distribution:

```
# Example Q-Q plot
set.seed(123) # For reproducibility
y <- 1:100
x <- rnorm(100) # Generating random normal data
qqplot(x,
       y,
       main = "Q-Q Plot",
       xlab = "Theoretical Quantiles",
       ylab = "Sample Quantiles")
```

5.1.2.5.2 Influential Observations and Outliers Identifying influential observations or outliers is essential for robust regression modeling. The **hat matrix** (**H**) plays a key role in diagnosing influence.

5.1.2.5.2.1 Hat Matrix: Outliers in X-Space The hat matrix is primarily concerned with **leverage**, which reflects how far an observation's predictor values (X-space) are from the centroid of the predictor space.

- **What it measures:** The diagonal elements of the hat matrix quantify **leverage**, not residual size or model fit. High leverage suggests that an observation has an unusual predictor configuration and might disproportionately influence the regression line, irrespective of the response variable.
- **What it doesn't measure:** It doesn't directly account for outliers in Y-space or residuals.

The hat matrix, defined as:

$$\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}$$

has the following properties:

- **Fitted Values**: $\hat{\mathbf{Y}} = \mathbf{HY}$.
- **Residuals**: $\mathbf{e} = (\mathbf{I} - \mathbf{H})\mathbf{Y}$.
- **Variance of Residuals**: $\text{var}(\mathbf{e}) = \sigma^2(\mathbf{I} - \mathbf{H})$.

Diagonal Elements of the Hat Matrix (h_{ii})

- h_{ii} is the i-th element on the main diagonal of **H**. It must satisfy $0 \leq h_{ii} \leq 1$.
- $\sum_{i=1}^{n} h_{ii} = p$, where p is the number of parameters (including the intercept).
- The variance of residuals for observation i is given by $\sigma^2(e_i) = \sigma^2(1 - h_{ii})$.
- The covariance between residuals for observations i and j ($i \neq j$) is $-h_{ij}\sigma^2$.

For large datasets, the off-diagonal elements (h_{ij}) tend to have small covariance if model assumptions hold.

Estimations Using MSE

- Variance of residuals: $s^2(e_i) = MSE(1 - h_{ii})$, where MSE is the mean squared error.
- Covariance of residuals: $\hat{\text{cov}}(e_i, e_j) = -h_{ij}(MSE)$.

Interpretation of h_{ii}

If $\mathbf{x}_i = [1, X_{i,1}, \dots, X_{i,p-1}]'$ represents the vector of predictor values for observation i, then:

$$h_{ii} = \mathbf{x}_i'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_i$$

The value of h_{ii} depends on the relative positions of the design points $X_{i,1}, \dots, X_{i,p-1}$. Observations with high h_{ii} are more influential and warrant closer inspection for leverage or outlier behavior.

5.1.2.5.2.2 Studentized Residuals: Outliers in Y-Space Residuals focus on discrepancies between observed (Y) and predicted ($\widehat{Y}$) values, helping to identify outliers in Y-space.

- **What they measure:**
  - Standardized or studentized residuals assess how far an observation's response is from the regression line, adjusted for variance.
  - **Externally studentized residuals** are more robust because they exclude the i-th observation when estimating variance.
  - Large studentized residuals (e.g., > 2 or > 3) indicate observations that are unusual in Y-space.
- **What they don't measure:** They do not consider leverage or the X-space configuration. A point with a large residual could have low leverage, making it less influential overall.

Studentized residuals, also known as standardized residuals, adjust for the variance of residuals by dividing the residuals by their standard error:

$$r_i = \frac{e_i}{s(e_i)}$$
$$r_i \sim N(0, 1),$$

where $s(e_i) = \sqrt{MSE(1 - h_{ii})}$, and r_i accounts for the varying variances of residuals. These residuals allow for a better comparison of model fit across observations.

- **Semi-Studentized Residuals**: In contrast, the semi-studentized residuals are defined as:

$$e_i^* = \frac{e_i}{\sqrt{MSE}}$$

This approach does not adjust for the heterogeneity in variances of residuals, as e_i^* assumes equal variance for all residuals.

To assess the influence of individual observations, we consider the model without a particular value. When the i-th observation is removed, the **deleted residual** is defined as:

$$\begin{aligned} d_i &= Y_i - \hat{Y}_{i(i)} \\ &= \frac{e_i}{1 - h_{ii}}, \end{aligned}$$

where Y_i is the actual observation, and $\hat{Y}_{i(i)}$ is the predicted value for the i-th observation, computed using the regression model fitted to the remaining $n-1$ observations. Importantly, we do not need to refit the regression model for each observation to compute d_i.

- As h_{ii} (leverage) increases, d_i also increases, indicating higher influence of the observation.

The variance of the deleted residual is given by:

$$s^2(d_i) = \frac{MSE_{(i)}}{1 - h_{ii}},$$

where $MSE_{(i)}$ is the mean squared error when the i-th case is omitted.

The **studentized deleted residual** accounts for variability and follows a t-distribution with $n - p - 1$ degrees of freedom:

$$t_i = \frac{d_i}{s(d_i)} = \frac{e_i}{\sqrt{MSE_{(i)}(1 - h_{ii})}},$$

where t_i helps identify outliers more effectively.

We can compute t_i without fitting the regression model multiple times. Using the relationship:

$$(n - p)MSE = (n - p - 1)MSE_{(i)} + \frac{e_i^2}{1 - h_{ii}},$$

we derive:

$$t_i = e_i \sqrt{\frac{n - p - 1}{SSE(1 - h_{ii}) - e_i^2}}.$$

This formulation avoids the need for recalculating $MSE_{(i)}$ explicitly for each case.

Outlying Y-observations are those with large studentized deleted residuals in absolute value. To handle multiple testing when there are many residuals, we use a Bonferroni-adjusted critical value:

$$t_{1-\alpha/2n;n-p-1},$$

where α is the desired significance level, n is the sample size, and p is the number of parameters in the model. Observations exceeding this threshold are flagged as potential outliers.

```
# Example R Code for Demonstrating Residual Diagnostics
set.seed(123) # For reproducibility

# Simulate some data
n <- 100
x <- rnorm(n, mean = 10, sd = 3)
y <- 5 + 2 * x + rnorm(n, mean = 0, sd = 2)

# Fit a linear regression model
model <- lm(y ~ x)


# Extract residuals, fitted values, and hat values
residuals <- resid(model)
hat_values <- lm.influence(model)$hat
mse <- mean(residuals ^ 2)
n <- length(residuals)
p <- length(coefficients(model)) # Number of parameters

# Compute studentized residuals
studentized_residuals <- residuals / sqrt(mse * (1 - hat_values))

# Compute deleted residuals
deleted_residuals <- residuals / (1 - hat_values)

# Compute studentized deleted residuals
studentized_deleted_residuals <-
    residuals * sqrt((n - p - 1)
                     / (sum(residuals ^ 2) * (1 - hat_values) - residuals ^ 2))

# Flag potential outliers using Bonferroni-adjusted critical value
alpha <- 0.05
bonferroni_threshold <- qt(1 - alpha / (2 * n), df = n - p - 1)
outliers <-
```

```r
    abs(studentized_deleted_residuals) > bonferroni_threshold

# Print results
results <- data.frame(
    Residuals                      = residuals,
    Hat_Values                     = hat_values,
    Studentized_Residuals          = studentized_residuals,
    Deleted_Residuals              = deleted_residuals,
    Studentized_Deleted_Residuals = studentized_deleted_residuals,
    Outlier                        = outliers
)

causalverse::nice_tab(head(results))
#>   Residuals Hat_Values Studentized_Residuals Deleted_Residuals
#> 1     -1.27       0.02                 -0.67             -1.29
#> 2      0.70       0.01                  0.36              0.70
#> 3     -0.12       0.04                 -0.07             -0.13
#> 4     -0.48       0.01                 -0.25             -0.49
#> 5     -1.68       0.01                 -0.88             -1.70
#> 6      0.30       0.04                  0.16              0.31
#>   Studentized_Deleted_Residuals Outlier
#> 1                         -0.66   FALSE
#> 2                          0.36   FALSE
#> 3                         -0.06   FALSE
#> 4                         -0.25   FALSE
#> 5                         -0.87   FALSE
#> 6                          0.15   FALSE

# Plot studentized deleted residuals for visualization
plot(
    studentized_deleted_residuals,
    main = "Studentized Deleted Residuals",
    xlab = "Observation",
    ylab = "Studentized Deleted Residuals",
    pch = 16,
    col = ifelse(outliers, "red", "black")
)
abline(
    h = c(-bonferroni_threshold, bonferroni_threshold),
    col = "blue",
    lty = 2
)
legend(
    "topright",
    legend = c("Potential Outliers", "Threshold"),
```

```
    col = c("red", "blue"),
    pch = c(16, NA),
    lty = c(NA, 2)
)
```

5.1.2.5.3 Identifying Influential Cases By **influential**, we refer to observations whose exclusion causes major changes in the fitted regression model. Note that not all outliers are influential.

Types of Influence Measures

- **Influence on Single Fitted Values**: DFFITS
- **Influence on All Fitted Values**: Cook's D
- **Influence on the Regression Coefficients**: DFBETAS

Measures like Cook's D, DFFITS, and DFBETAS combine leverage (from the hat matrix) and residual size (from studentized residuals) to assess the **influence** of an observation on the model as a whole. Hence, these effectively combine impact of X-space and Y-space.

5.1.2.5.3.1 DFFITS DFFITS measures the **influence on single fitted values**. It is defined as:

$$
\begin{aligned}
(DFFITS)_i &= \frac{\hat{Y}_i - \hat{Y}_{i(i)}}{\sqrt{MSE_{(i)} h_{ii}}} \\
&= t_i \left(\frac{h_{ii}}{1 - h_{ii}} \right)^{1/2}
\end{aligned}
$$

Where:

- $\hat{Y}_i$ = fitted value for observation i using all data.
- $\hat{Y}_{i(i)}$ = fitted value for observation i with the ith case removed.
- $MSE_{(i)}$ = mean squared error with observation i excluded.
- h_{ii} = leverage of the ith observation.
- t_i = studentized deleted residual.

High DFFITS values occur when leverage and residuals are jointly significant.

DFFITS captures the **standardized difference between the fitted value for observation** i with and without the ith case in the model. It is a product of:

1. The **studentized deleted residual**.
2. A scaling factor based on the leverage of the ith observation, h_{ii}.

An observation is considered influential based on DFFITS if:

- **Small to medium data sets**: $|DFFITS| > 1$
- **Large data sets**: $|DFFITS| > 2\sqrt{p/n}$

Where:

- p = number of predictors (including the intercept).
- n = total number of observations.

This provides a practical threshold for detecting influential observations in different dataset sizes.

```r
# Load necessary package
library(car)

# Fit a linear model
model <- lm(mpg ~ hp + wt, data = mtcars)

# Compute DFFITS
dffits_values <- dffits(model)

# Display influential observations based on the threshold for a large dataset
threshold <- 2 * sqrt(length(coefficients(model)) / nrow(mtcars))
influential_obs <- which(abs(dffits_values) > threshold)
```

```r
# Results
list(
    DFFITS                   = dffits_values,
    Threshold                = threshold,
    Influential_Observations = influential_obs
)
#> $DFFITS
#>           Mazda RX4       Mazda RX4 Wag          Datsun 710      Hornet 4 Drive
#>        -0.218494101        -0.126664789        -0.249103400         0.011699160
#>   Hornet Sportabout             Valiant          Duster 360           Merc 240D
#>         0.028162679        -0.253806124        -0.191618944         0.221917842
#>            Merc 230            Merc 280           Merc 280C          Merc 450SE
#>         0.079763706        -0.067222732        -0.190099538         0.064280875
#>          Merc 450SL         Merc 450SLC  Cadillac Fleetwood Lincoln Continental
#>         0.020560728        -0.135714533         0.008984366         0.227919348
#>   Chrysler Imperial            Fiat 128         Honda Civic      Toyota Corolla
#>         1.231668760         0.749153703         0.165329646         0.865985851
#>       Toyota Corona    Dodge Challenger         AMC Javelin          Camaro Z28
#>        -0.292008465        -0.253389811        -0.294709853        -0.170476763
#>    Pontiac Firebird           Fiat X1-9       Porsche 914-2        Lotus Europa
#>         0.207813200        -0.041423665        -0.004054382         0.471518032
#>      Ford Pantera L        Ferrari Dino       Maserati Bora          Volvo 142E
#>        -0.161026362        -0.129395315         0.907521354        -0.128232538
#>
#> $Threshold
#> [1] 0.6123724
#>
#> $Influential_Observations
#> Chrysler Imperial          Fiat 128    Toyota Corolla     Maserati Bora
#>                17                18                20                31
```

5.1.2.5.3.2 Cook's D **Cook's D** measures the **influence of the** ith case on all fitted values in a regression model. It is defined as:

$$
\begin{aligned}
D_i &= \frac{\sum_{j=1}^{n}(\hat{Y}_j - \hat{Y}_{j(i)})^2}{p(MSE)} \\
&= \frac{e_i^2}{p(MSE)}\left(\frac{h_{ii}}{(1-h_{ii})^2}\right)
\end{aligned}
$$

Where:

- $\hat{Y}_j$ = fitted value for observation j using all data.
- $\hat{Y}_{j(i)}$ = fitted value for observation j with the ith case removed.
- e_i = residual for observation i.

- h_{ii} = leverage of the ith observation.
- p = number of predictors (including the intercept).
- MSE = mean squared error of the model.

Key Insights

- **Cook's D quantifies the overall influence** of the ith observation on the entire set of fitted values.
- If either the residual e_i increases or the leverage h_{ii} increases, then D_i also increases, indicating higher influence.
- Observations with both high leverage and large residuals are flagged as influential.

Threshold for Influence

- D_i can be interpreted as a percentile of an $F_{(p,n-p)}$ distribution.
- Practical thresholds:
  - **If** $D_i > 4/n$, the ith case has major influence.
  - Alternatively, cases where D_i exceeds the 50th percentile of the F-distribution may also be considered influential.

```r
# Load necessary package
library(car)

# Fit a linear model
model <- lm(mpg ~ hp + wt, data = mtcars)

# Compute Cook's D
cooks_d_values <- cooks.distance(model)

# Display influential observations based on the threshold
threshold <- 4 / nrow(mtcars)
influential_obs <- which(cooks_d_values > threshold)

# Results
list(
  Cooks_D                  = cooks_d_values,
  Threshold                = threshold,
  Influential_Observations = influential_obs
)
#> $Cooks_D
#>           Mazda RX4       Mazda RX4 Wag          Datsun 710      Hornet 4 Drive
#>        1.589652e-02        5.464779e-03        2.070651e-02        4.724822e-05
#>   Hornet Sportabout             Valiant          Duster 360           Merc 240D
#>        2.736184e-04        2.155064e-02        1.255218e-02        1.677650e-02
#>            Merc 230            Merc 280           Merc 280C          Merc 450SE
#>        2.188702e-03        1.554996e-03        1.215737e-02        1.423008e-03
#>          Merc 450SL         Merc 450SLC  Cadillac Fleetwood Lincoln Continental
```

```
#>        1.458960e-04        6.266049e-03        2.786686e-05        1.780910e-02
#>   Chrysler Imperial            Fiat 128         Honda Civic      Toyota Corolla
#>        4.236109e-01        1.574263e-01        9.371446e-03        2.083933e-01
#>       Toyota Corona    Dodge Challenger         AMC Javelin          Camaro Z28
#>        2.791982e-02        2.087419e-02        2.751510e-02        9.943527e-03
#>    Pontiac Firebird           Fiat X1-9       Porsche 914-2        Lotus Europa
#>        1.443199e-02        5.920440e-04        5.674986e-06        7.353985e-02
#>      Ford Pantera L        Ferrari Dino       Maserati Bora          Volvo 142E
#>        8.919701e-03        5.732672e-03        2.720397e-01        5.600804e-03
#>
#> $Threshold
#> [1] 0.125
#>
#> $Influential_Observations
#> Chrysler Imperial          Fiat 128    Toyota Corolla     Maserati Bora
#>                17                18                20                31
```

5.1.2.5.3.3 DFBETAS **DFBETAS** measures the **influence of the** ith observation on each regression coefficient in a regression model. It is defined as:

$$(DFBETAS)_{k(i)} = \frac{b_k - b_{k(i)}}{\sqrt{MSE_{(i)} c_{kk}}}$$

Where:

- b_k = regression coefficient for the kth predictor using all observations.
- $b_{k(i)}$ = regression coefficient for the kth predictor with the ith observation omitted.
- $MSE_{(i)}$ = mean squared error with the ith observation excluded.
- c_{kk} = kth diagonal element of $\mathbf{X}'\mathbf{X}^{-1}$, representing the variance of b_k.

Key Insights

- **DFBETAS quantifies the change in each regression coefficient** (b_k) caused by omitting the ith observation.
- The sign of DFBETAS indicates whether the inclusion of an observation increases or decreases the regression coefficient.
  - Positive DFBETAS: Inclusion increases b_k.
  - Negative DFBETAS: Inclusion decreases b_k.
- High DFBETAS indicate that a single observation disproportionately affects one or more predictors.

The thresholds for identifying influential observations based on DFBETAS are:

- **Small data sets**: $|DFBETAS| > 1$
- **Large data sets**: $|DFBETAS| > 2/\sqrt{n}$

Where n is the total number of observations.

```r
# Load necessary package
library(car)

# Fit a linear model
model <- lm(mpg ~ hp + wt, data = mtcars)

# Compute DFBETAS
dfbetas_values <- dfbetas(model)

# Display influential observations based on the threshold for each predictor
threshold <- 2 / sqrt(nrow(mtcars))
influential_obs <-
    apply(dfbetas_values, 2, function(x)
        which(abs(x) > threshold))

# Results
list(
    DFBETAS                  = dfbetas_values,
    Threshold                = threshold,
    Influential_Observations = influential_obs
)
#> $DFBETAS
#>                       (Intercept)           hp             wt
#> Mazda RX4           -0.161347204  0.032966471  0.0639304305
#> Mazda RX4 Wag       -0.069324050  0.045785122 -0.0004066495
#> Datsun 710          -0.211199646  0.043374926  0.0972314374
#> Hornet 4 Drive       0.002672687 -0.006839301  0.0044886906
#> Hornet Sportabout    0.001784844  0.009208434 -0.0015536931
#> Valiant             -0.005985946  0.180374447 -0.1516565139
#> Duster 360           0.004705177 -0.159988770  0.0781031774
#> Merc 240D            0.034255292 -0.189552940  0.1224118752
#> Merc 230             0.019788247 -0.055075623  0.0332570461
#> Merc 280            -0.003198686  0.036709039 -0.0337297820
#> Merc 280C           -0.009045583  0.103809696 -0.0953846390
#> Merc 450SE          -0.026973686 -0.005712458  0.0356973740
#> Merc 450SL          -0.003961562  0.003399822  0.0049302300
#> Merc 450SLC          0.031572445 -0.016800308 -0.0400515832
#> Cadillac Fleetwood  -0.006420656 -0.002577897  0.0075499557
#> Lincoln Continental -0.168791258 -0.058242601  0.1903129995
#> Chrysler Imperial   -0.924056752 -0.148009806  0.9355996760
#> Fiat 128             0.605181396 -0.311246566 -0.1672758566
#> Honda Civic          0.156388333 -0.034026915 -0.0819144214
#> Toyota Corolla       0.804669969 -0.170934240 -0.4114605894
#> Toyota Corona       -0.231328587  0.066064464  0.0882138248
```

```
#> Dodge Challenger     0.003923967  0.049775308 -0.0888481611
#> AMC Javelin         -0.019610048  0.037837437 -0.0734203131
#> Camaro Z28           0.029920076 -0.128670440  0.0390740055
#> Pontiac Firebird    -0.058806962 -0.002278294  0.0868742949
#> Fiat X1-9           -0.037559007  0.010208853  0.0174261386
#> Porsche 914-2       -0.003655931  0.000316321  0.0020588013
#> Lotus Europa         0.423409344  0.188396749 -0.4072338373
#> Ford Pantera L      -0.022536462 -0.148176049  0.0999346699
#> Ferrari Dino        -0.065508308 -0.085182962  0.0869804902
#> Maserati Bora       -0.007482815  0.865763737 -0.4999048760
#> Volvo 142E          -0.080001907  0.038406565  0.0127537553
#>
#> $Threshold
#> [1] 0.3535534
#>
#> $Influential_Observations
#> $Influential_Observations$`(Intercept)`
#> Chrysler Imperial          Fiat 128    Toyota Corolla      Lotus Europa
#>                17                18                20                28
#>
#> $Influential_Observations$hp
#> Maserati Bora
#>            31
#>
#> $Influential_Observations$wt
#> Chrysler Imperial    Toyota Corolla      Lotus Europa     Maserati Bora
#>                17                20                28                31
```

5.1.2.5.4 Collinearity **Collinearity** (or multicollinearity) refers to the correlation among explanatory variables in a regression model. It can lead to various issues, including:

- Large changes in the estimated regression coefficients when a predictor variable is added or removed, or when observations are altered.
- Non-significant results for individual tests on regression coefficients of important predictor variables.
- Regression coefficients with signs opposite to theoretical expectations or prior experience.
- Large coefficients of simple correlation between pairs of predictor variables in the correlation matrix.
- Wide confidence intervals for the regression coefficients representing important predictor variables.

When some X variables are highly correlated, the inverse $(X'X)^{-1}$ either does not exist or is computationally unstable. This can result in:

1. **Non-interpretability of parameters**:

$$\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

2. **Infinite sampling variability**:

$$\mathbf{s^2}(\mathbf{b}) = \mathbf{MSE}(\mathbf{X}'\mathbf{X})^{-1}$$

If some predictor variables (X) are "perfectly" correlated, the system becomes undetermined, leading to an infinite number of models that fit the data. Specifically:

- If $X'X$ is singular, then $(X'X)^{-1}$ does not exist.
- This results in poor parameter estimation and invalid statistical inference.

5.1.2.5.4.1 Variance Inflation Factors (VIFs) The **Variance Inflation Factor (VIF)** quantifies how much the variance of a regression coefficient is inflated due to multicollinearity. It is defined as:

$$VIF_k = \frac{1}{1 - R_k^2}$$

Where:

- R_k^2 is the coefficient of multiple determination when X_k is regressed on the other $p - 2$ predictor variables in the model.

Interpretation of VIFs

- Large VIF_k values indicate that near collinearity is inflating the variance of b_k. The relationship is given by:

$$var(b_k) \propto \sigma^2(VIF_k)$$

- **Thresholds**:
  - $VIF > 4$: Investigate the cause of multicollinearity.
  - $VIF_k > 10$: Serious multicollinearity problem that can lead to poor parameter estimates.
- The **mean VIF** provides an estimate of the degree of multicollinearity:
  - If $avg(VIF) >> 1$, serious multicollinearity is present.
- **Multicollinearity and VIF:**
  - High VIFs with indicator variables are normal and not problematic.
  - VIF is generally not useful for detecting multicollinearity concerns in models with fixed effects.
- **Overemphasis on Multicollinearity:**

  - Multicollinearity inflates standard errors and widens confidence intervals but does not bias results.
  - If key variables have narrow confidence intervals, multicollinearity is not an issue.
- **Goldberger's Insight** (Goldberger, 1991)**:**
  - Multicollinearity is akin to small sample size ("micronumerosity").
  - Large standard errors are expected with highly correlated independent variables.
- **Practical Implications:**
  - Evaluate whether confidence intervals for key variables are sufficiently narrow.
  - If not, the study is inconclusive, and a larger dataset or redesigned study is needed.

```r
# Load necessary packages
library(car)

# Fit a regression model
model <- lm(mpg ~ hp + wt + disp, data = mtcars)

# Compute Variance Inflation Factors
vif_values <- vif(model)

# Check for high multicollinearity
threshold <- 10
high_vif <- which(vif_values > threshold)

# Results
list(
  VIFs = vif_values,
  High_VIF_Threshold = threshold,
  High_VIF_Indices = high_vif
)
#> $VIFs
#>       hp       wt     disp
#> 2.736633 4.844618 7.324517
#>
#> $High_VIF_Threshold
#> [1] 10
#>
#> $High_VIF_Indices
#> named integer(0)
```

5.1.2.5.4.2 Condition Number **Condition Number** is a diagnostic measure for detecting multicollinearity, derived from the spectral decomposition of the matrix $\mathbf{X}'\mathbf{X}$.

The spectral decomposition of $\mathbf{X}'\mathbf{X}$ is:

$$\mathbf{X}'\mathbf{X} = \sum_{i=1}^{p} \lambda_i \mathbf{u_i} \mathbf{u_i'}$$

Where:

- λ_i: Eigenvalue associated with the ith eigenvector.
- $\mathbf{u}_i$: Eigenvector associated with λ_i.
- $\lambda_1 > \lambda_2 > \cdots > \lambda_p$ (ordered eigenvalues).
- The eigenvectors are orthogonal:

$$\begin{cases} \mathbf{u_i'u_j} = 0 & \text{for } i \neq j \\ \mathbf{u_i'u_j} = 1 & \text{for } i = j \end{cases}$$

Definition of the Condition Number

The **Condition Number** is defined as:

$$k = \sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}}$$

Where:

- $\lambda_{\max}$: Largest eigenvalue.
- $\lambda_{\min}$: Smallest eigenvalue.

Interpretation

- $k > 30$: Cause for concern.
- $30 < k < 100$: Moderate dependencies among predictors.
- $k > 100$: Strong collinearity, indicating serious multicollinearity.

The **Condition Index** for the ith eigenvalue is defined as:

$$\delta_i = \sqrt{\frac{\lambda_{\max}}{\lambda_i}}$$

Where $i = 1, \ldots, p$.

Variance proportions can be used to identify collinearity issues. The proportion of total variance associated with the kth regression coefficient and the ith eigen mode is given by:

$$\frac{u_{ik}^2/\lambda_i}{\sum_j \left(u_{jk}^2/\lambda_j\right)}$$

Key Indicators:

- A **large condition index** δ_i suggests potential collinearity.
- Variance proportions > 0.5 for at least two regression coefficients indicate serious collinearity problems.

```r
# Load necessary package
library(car)

# Fit a regression model
model <- lm(mpg ~ hp + wt + disp, data = mtcars)

# Compute eigenvalues and eigenvectors of the correlation matrix of predictors
cor_matrix <- cor(mtcars[, c("hp", "wt", "disp")])
eigen_decomp <- eigen(cor_matrix)

# Extract eigenvalues and compute the Condition Number
eigenvalues <- eigen_decomp$values
condition_number <- sqrt(max(eigenvalues) / min(eigenvalues))

# Compute Condition Indices
condition_indices <- sqrt(max(eigenvalues) / eigenvalues)

# Results
list(
  Condition_Number = condition_number,
  Condition_Indices = condition_indices
)
#> $Condition_Number
#> [1] 5.469549
#>
#> $Condition_Indices
#> [1] 1.000000 2.697266 5.469549
```

1. **Condition Number**: A single value indicating the degree of multicollinearity.
2. **Condition Indices**: A vector showing the relative dependency associated with each eigenvalue.

5.1.2.5.5 Constancy of Error Variance Testing for the constancy of error variance (homoscedasticity) ensures that the assumptions of linear regression are met.

Below are two commonly used tests to assess error variance.

5.1.2.5.5.1 Brown-Forsythe Test (Modified Levene Test) The **Brown-Forsythe Test** does not depend on the normality of errors and is suitable when error variance increases or decreases with X.

Procedure

1. Split the residuals into two groups:

$$e_{i1}, i = 1, \dots, n_1 \quad \text{and} \quad e_{j2}, j = 1, \dots, n_2$$

2. Compute absolute deviations from the group medians:

$$d_{i1} = |e_{i1} - \tilde{e}_1| \quad \text{and} \quad d_{j2} = |e_{j2} - \tilde{e}_2|$$

   where $\tilde{e}_1$ and $\tilde{e}_2$ are the medians of groups 1 and 2, respectively.

3. Perform a two-sample t-test on d_{i1} and d_{j2}:

$$t_L = \frac{\bar{d}_1 - \bar{d}_2}{s\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

   where

$$s^2 = \frac{\sum_i (d_{i1} - \bar{d}_1)^2 + \sum_j (d_{j2} - \bar{d}_2)^2}{n - 2}$$

4. Reject the null hypothesis of constant error variance if:

$$|t_L| > t_{1-\alpha/2;n-2}$$

```r
# Load necessary package
library(car)

# Fit a linear model
model <- lm(mpg ~ hp + wt, data = mtcars)

# Perform the Brown-Forsythe Test
levene_test <- leveneTest(model$residuals ~ cut(mtcars$hp, 2))  # Split HP into 2 groups

# Results
levene_test
```

```
#> Levene's Test for Homogeneity of Variance (center = median)
#>       Df F value Pr(>F)
#> group  1  0.0851 0.7724
#>       30
```

The p-value determines whether to reject the null hypothesis of constant variance.

5.1.2.5.5.2 Breusch-Pagan Test (Cook-Weisberg Test) The **Breusch-Pagan Test** assumes independent and normally distributed errors. It tests the hypothesis:

- $H_0 : \gamma_1 = 0$ (Constant variance).
- $H_1 : \gamma_1 \neq 0$ (Non-constant variance).

Procedure

1. Assume the variance of the error terms is modeled as:

$$\sigma_i^2 = \gamma_0 + \gamma_1 X_i$$

2. Regress the squared residuals (e_i^2) on X_i:
   - Obtain the **regression sum of squares** (SSR^*).
3. Compute the Breusch-Pagan statistic:

$$X_{BP}^2 = \frac{SSR^*/2}{\left(\frac{SSE}{n}\right)^2}$$

   where SSE is the error sum of squares from the regression of Y on X.
4. Compare X_{BP}^2 to the critical value from the χ^2 distribution with 1 degree of freedom:
   - Reject H_0 (homogeneous variance) if:

$$X_{BP}^2 > \chi^2_{1-\alpha;1}$$

```r
# Load necessary package
library(lmtest)

# Perform the Breusch-Pagan Test
bp_test <- bptest(model)

# Results
bp_test
#>
```

```
#>  studentized Breusch-Pagan test
#>
#> data:  model
#> BP = 0.88072, df = 2, p-value = 0.6438
```

If the p-value is below the chosen significance level (e.g., 0.05), reject H_0 and conclude non-constant variance.

5.1.2.5.6 Independence Testing for independence ensures that the residuals of a regression model are uncorrelated. Violation of this assumption may lead to biased or inefficient parameter estimates. Below, we discuss three primary methods for diagnosing dependence: plots, the Durbin-Watson test, and specific methods for time-series and spatial data.

5.1.2.5.6.1 Plots A residual plot can help detect dependence in the residuals:

1. Plot residuals (e_i) versus the predicted values ($\hat{Y}_i$).
2. Patterns such as systematic waves, trends, or clusters indicate possible dependence.
3. Independence is suggested if residuals are randomly scattered without clear patterns.

```r
# Fit a regression model
model <- lm(mpg ~ hp + wt, data = mtcars)

# Residual plot
plot(model$fitted.values, model$residuals,
     xlab = "Fitted Values",
     ylab = "Residuals",
     main = "Residual Plot")
abline(h = 0, col = "red")
```

5.1.2.5.6.2 Durbin-Watson Test The **Durbin-Watson test** specifically detects autocorrelation in residuals, especially in time-series data.

- **Hypotheses**:
  - H_0: Residuals are uncorrelated.
  - H_1: Residuals are autocorrelated.
- The Durbin-Watson statistic (d) is calculated as: $d = \frac{\sum_{t=2}^{n}(e_t - e_{t-1})^2}{\sum_{t=1}^{n} e_t^2}$
  - $d \approx 2$: No autocorrelation.
  - $d < 2$: Positive autocorrelation.
  - $d > 2$: Negative autocorrelation.

```
# Load necessary package
library(lmtest)

# Perform the Durbin-Watson test
dw_test <- dwtest(model)

# Results
dw_test
#>
#>  Durbin-Watson test
#>
#> data:  model
```

```
#> DW = 1.3624, p-value = 0.02061
#> alternative hypothesis: true autocorrelation is greater than 0
```

5.1.2.5.6.3 Time-Series Autocorrelation For time-series data, autocorrelation often occurs due to the temporal structure. Key diagnostics include:

1. **Autocorrelation Function (ACF)**:
   - Shows the correlation of residuals with their lagged values.
   - Significant spikes in the ACF plot indicate autocorrelation.
2. **Partial Autocorrelation Function (PACF)**:
   - Identifies the correlation of residuals after removing the influence of intermediate lags.

```r
# Load necessary package
library(forecast)

# Residuals from a time-series model
time_series_res <- ts(model$residuals)

# Plot ACF and PACF
acf(time_series_res, main = "ACF of Residuals")
```

```r
pacf(time_series_res, main = "PACF of Residuals")
```

5.1.2.5.6.4 Spatial Statistics Spatial dependence occurs when residuals are correlated across geographical or spatial locations. Two primary tests are used to diagnose spatial autocorrelation:

Moran's I measures global spatial autocorrelation. It determines whether similar values cluster spatially. The statistic is defined as:

$$I = \frac{n}{W} \cdot \frac{\sum_i \sum_j w_{ij}(x_i - \bar{x})(x_j - \bar{x})}{\sum_i (x_i - \bar{x})^2}$$

Where:

- n: Number of observations.
- W: Sum of all spatial weights w_{ij}.
- w_{ij}: Spatial weight between locations i and j.
- x_i: Residual value at location i.
- $\bar{x}$: Mean of the residuals.

Spatial Weight Matrix (W): W defines the spatial relationship between observations. It is often derived from:

- **Distance-based methods**: E.g., k-nearest neighbors or distance bands.
- **Adjacency methods**: Based on shared boundaries in geographic data.

Interpretation:

- $I > 0$: Positive spatial autocorrelation (similar values cluster together).

- $I < 0$: Negative spatial autocorrelation (dissimilar values are neighbors).

- $I \approx 0$: Random spatial distribution.

```r
# Load necessary packages
library(spdep)

# Simulate spatial data (example with mtcars dataset)
coords <- cbind(mtcars$hp, mtcars$wt)  # Coordinates based on two variables
neighbors <- knn2nb(knearneigh(coords, k = 3))  # Nearest neighbors
weights <- nb2listw(neighbors, style = "W")  # Spatial weights

# Compute Moran's I for residuals
model <- lm(mpg ~ hp + wt, data = mtcars)
moran_test <- moran.test(model$residuals, weights)

# Results
moran_test
#>
#>  Moran I test under randomisation
#>
#> data:  model$residuals
#> weights: weights
#>
#> Moran I statistic standard deviate = 1.5146, p-value = 0.06494
#> alternative hypothesis: greater
#> sample estimates:
#> Moran I statistic       Expectation          Variance
#>        0.15425069       -0.03225806        0.01516371

# Moran's scatterplot
moran.plot(model$residuals, weights, main = "Moran's Scatterplot")
```

Significant Moran's I: Indicates global clustering of similar residuals, suggesting spatial dependence.

Geary's C measures spatial autocorrelation at a local level, emphasizing differences between neighboring observations. The statistic is defined as:

$$C = \frac{n-1}{2W} \cdot \frac{\sum_i \sum_j w_{ij}(x_i - x_j)^2}{\sum_i (x_i - \bar{x})^2}$$

Where:

- C ranges from 0 to 2:
  - $C \approx 0$: High positive spatial autocorrelation.
  - $C \approx 1$: Spatial randomness.
  - $C \approx 2$: High negative spatial autocorrelation.

Comparison of Moran's I and Geary's C:

- Moran's I is more global and measures the overall pattern of autocorrelation.
- Geary's C is more sensitive to local spatial autocorrelation, detecting small-scale patterns.

```
# Compute Geary's C for residuals
geary_test <- geary.test(model$residuals, weights)
```

```
# Results
geary_test
#>
#>  Geary C test under randomisation
#>
#> data:  model$residuals
#> weights: weights
#>
#> Geary C statistic standard deviate = 1.0592, p-value = 0.1447
#> alternative hypothesis: Expectation greater than statistic
#> sample estimates:
#> Geary C statistic       Expectation          Variance
#>        0.84535708        1.00000000        0.02131399
```

Significant Geary's C: Highlights local spatial autocorrelation, useful for identifying specific regions or groups of observations where dependence is strong.

5.2 Generalized Least Squares

5.2.1 Infeasible Generalized Least Squares

Motivation for a More Efficient Estimator

- The Gauss-Markov Theorem guarantees that OLS is the **Best Linear Unbiased Estimator (BLUE)** under assumptions A1-A4:
  - **A4:** $Var(\epsilon|\mathbf{X}) = \sigma^2\mathbf{I}_n$ (homoscedasticity and no autocorrelation).
- When **A4** does not hold:
  - **Heteroskedasticity:** $Var(\epsilon_i|\mathbf{X}) \neq \sigma^2$.
  - **Serial Correlation:** $Cov(\epsilon_i, \epsilon_j|\mathbf{X}) \neq 0$ for $(i \neq j)$.

Without **A4**, OLS is **unbiased** but no longer **efficient**. This motivates the need for an alternative approach to identify the most efficient estimator.

The unweighted (standard) regression model is given by:

$$\mathbf{y} = \mathbf{X} \ + \epsilon$$

Assuming **A1-A3** hold (linearity, full rank, exogeneity), but **A4** does not, the variance of the error term is no longer proportional to an identity

$$Var(\epsilon|\mathbf{X}) = \Omega \neq \sigma^2\mathbf{I}_n.$$

To address the violation of **A4** ($\Omega \neq \sigma^2\mathbf{I}_n$), one can transform the model by pre-multiplying both sides by a full-rank matrix $\mathbf{w}$ to have a weighted (transformed) regression model:

$$\mathbf{wy} = \mathbf{wX} \ + \mathbf{w} \ ,$$

where $\mathbf{w}$ is a full-rank matrix chosen such that:

$$\mathbf{w}'\mathbf{w} = \Omega^{-1}.$$

- $\mathbf{w}$ is the Cholesky Decomposition of Ω^{-1}.
- The Cholesky decomposition ensures $\mathbf{w}$ satisfies $\mathbf{w}'\mathbf{w} = \ ^{\mathbf{-1}}$, where $\mathbf{w}$ is the "square root" of Ω^{-1} in matrix terms.

By transforming the original model, the variance of the transformed errors becomes:

$$\Omega = Var(\epsilon|\mathbf{X}),$$
$$\Omega^{-1} = Var(\epsilon|\mathbf{X})^{-1}.$$

The transformed equation allows us to compute a more efficient estimator.

Using the transformed model, the **Infeasible Generalized Least Squares (IGLS)** estimator is:

$$\begin{aligned}
\hat{\ }_{\mathbf{IGLS}} &= (\mathbf{X}'\mathbf{w}'\mathbf{wX})^{\mathbf{-1}}\mathbf{X}'\mathbf{w}'\mathbf{wy} \\
&= (\mathbf{X}'\Omega^{\mathbf{-1}}\mathbf{X})^{\mathbf{-1}}\mathbf{X}'\Omega^{\mathbf{-1}}\mathbf{y} \\
&= \ + (\mathbf{X}'\Omega^{\mathbf{-1}}\mathbf{X})^{\mathbf{-1}}\mathbf{X}'\Omega^{\mathbf{-1}}\epsilon.
\end{aligned}$$

1. **Unbiasedness**

Since assumptions **A1-A3** hold for the unweighted model:

$$\begin{aligned}
\mathbf{E}(\hat{\ }_{\mathbf{IGLS}}|\mathbf{X}) &= \mathbf{E}(\ + (\mathbf{X}'\ ^{\mathbf{-1}}\mathbf{X}'\ ^{\mathbf{-1}}\ )|\mathbf{X}) \\
&= \ + \mathbf{E}(\mathbf{X}'\ ^{\mathbf{-1}}\mathbf{X}'\ ^{\mathbf{-1}}\ |\mathbf{X}) \\
&= \ + \mathbf{X}'\ ^{\mathbf{-1}}\mathbf{X}'\ ^{\mathbf{-1}}\mathbf{E}(\ |\mathbf{X}) \qquad \text{since A3: } E(\epsilon|\mathbf{X}) = 0, \\
&= \ .
\end{aligned}$$

Thus, the **IGLS estimator is unbiased**.

2. **Variance**

The variance of the transformed errors is given by:

$$\begin{aligned}
\mathbf{Var}(\mathbf{w}\ |\mathbf{X}) &= \mathbf{wVar}(\ |\mathbf{X})\mathbf{w}' \\
&= \mathbf{w}\ \mathbf{w}' \\
&= \mathbf{w}(\mathbf{w}'\mathbf{w})^{\mathbf{-1}}\mathbf{w}' \qquad \text{since } \mathbf{w} \text{ is full-rank,} \\
&= \mathbf{ww}^{\mathbf{-1}}(\mathbf{w}')^{\mathbf{-1}}\mathbf{w}' \\
&= \mathbf{I_n}.
\end{aligned}$$

Hence, **A4** holds for the transformed (weighted) equation, satisfying the Gauss-Markov conditions.

The variance of the IGLS estimator is:

$$\begin{aligned}
\mathbf{Var}(\hat{\ }_{\mathbf{IGLS}}|\mathbf{X}) &= \mathbf{Var}(\ + (\mathbf{X}'\ ^{-1}\mathbf{X})^{-1}\mathbf{X}'\ ^{-1}\ |\mathbf{X}) \\
&= \mathbf{Var}((\mathbf{X}'\ ^{-1}\mathbf{X})^{-1}\mathbf{X}'\ ^{-1}\ |\mathbf{X}) \\
&= (\mathbf{X}'\ ^{-1}\mathbf{X})^{-1}\mathbf{X}'\ ^{-1}\mathbf{Var}(\ |\mathbf{X})\ ^{-1}\mathbf{X}(\mathbf{X}'\ ^{-1}\mathbf{X})^{-1} \quad \text{because A4 holds,} \\
&= (\mathbf{X}'\ ^{-1}\mathbf{X})^{-1}\mathbf{X}'\ ^{-1}\ \ ^{-1}\ ^{-1}\mathbf{X}(\mathbf{X}'\ ^{-1}\mathbf{X})^{-1}, \\
&= (\mathbf{X}'\ ^{-1}\mathbf{X})^{-1}.
\end{aligned}$$

3. **Efficiency**

The difference in variances between OLS and IGLS is:

$$\mathbf{Var}(\hat{\ }_{\mathbf{OLS}}|\mathbf{X}) - \mathbf{Var}(\hat{\ }_{\mathbf{IGLS}}|\mathbf{X}) = \mathbf{A}\ \mathbf{A}',$$

where:

$$\mathbf{A} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' - (\mathbf{X}'\ ^{-1}\mathbf{X})^{-1}\mathbf{X}'\ ^{-1}.$$

Since is positive semi-definite, $\mathbf{A}\ \mathbf{A}'$ is also positive semi-definite. Thus, the **IGLS estimator is more efficient than OLS** under heteroskedasticity or autocorrelation.

In short, properties of $\hat{\ }_{\mathbf{IGLS}}$:

1. **Unbiasedness**: $\hat{\ }_{\mathbf{IGLS}}$ remains unbiased as long as **A1-A3** hold.
2. **Efficiency**: $\hat{\ }_{\mathbf{IGLS}}$ is more efficient than OLS under heteroskedasticity or serial correlation since it accounts for the structure of Ω.

Why Is IGLS "Infeasible"?

The name **infeasible** arises because it is generally impossible to compute the estimator directly due to the structure of $\mathbf{w}$ (or equivalently Ω^{-1}). The matrix $\mathbf{w}$ is defined as:

$$\mathbf{w} = \begin{pmatrix} w_{11} & 0 & 0 & \cdots & 0 \\ w_{21} & w_{22} & 0 & \cdots & 0 \\ w_{31} & w_{32} & w_{33} & \cdots & \cdots \\ w_{n1} & w_{n2} & w_{n3} & \cdots & w_{nn} \end{pmatrix},$$

with $n(n+1)/2$ unique elements for n observations. This results in more parameters than data points, making direct estimation infeasible.

To make the estimation feasible, assumptions about the structure of are required. Common approaches include:

- Heteroskedasticity Errors: Assume a multiplicative exponential model for the variance, such as $Var(\epsilon_i|\mathbf{X}) = \sigma_i^2$.
  - Assume no correlation between errors, but allow heterogeneous variances:
$$= \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n^2 \end{pmatrix}.$$
  - Estimate σ_i^2 using methods such as:
    * Modeling σ_i^2 as a function of predictors (e.g., $\sigma_i^2 = \exp(\mathbf{x}_i\gamma)$).
- Serial Correlation: Assume serial correlation follows an autoregressive process AR(1) Model, e.g., $\epsilon_t = \rho\epsilon_{t-1} + u_t$ and $Cov(\epsilon_t, \epsilon_{t-h}) = \rho^h\sigma^2$, where we have a variance-covariance matrix with off-diagonal elements decaying geometrically:
$$= \frac{\sigma^2}{1-\rho^2} \begin{pmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{n-1} \\ \rho & 1 & \rho & \cdots & \rho^{n-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{n-1} & \rho^{n-2} & \rho^{n-3} & \cdots & 1 \end{pmatrix}.$$
- Cluster Errors: Assume block-diagonal structure for to account for grouped or panel data.

Each assumption simplifies the estimation of and thus $\mathbf{w}$, enabling Feasible Generalized Least Squares with fewer unknown parameters to estimate.

5.2.2 Feasible Generalized Least Squares

5.2.2.1 Heteroskedasticity Errors

Heteroskedasticity occurs when the variance of the error term is not constant across observations. Specifically:

$$Var(\epsilon_i|x_i) = E(\epsilon_i^2|x_i) \neq \sigma^2,$$

but instead depends on a function of x_i:

$$Var(\epsilon_i|x_i) = h(x_i) = \sigma_i^2$$

This violates the assumption of homoscedasticity (constant variance), impacting the efficiency of OLS estimates.

For the model:

$$y_i = x_i\beta + \epsilon_i,$$

we apply a transformation to standardize the variance:

$$\frac{y_i}{\sigma_i} = \frac{x_i}{\sigma_i}\beta + \frac{\epsilon_i}{\sigma_i}.$$

By scaling each observation with $1/\sigma_i$, the variance of the transformed error term becomes:

$$\begin{aligned} Var\left(\frac{\epsilon_i}{\sigma_i}\middle| X\right) &= \frac{1}{\sigma_i^2} Var(\epsilon_i|X) \\ &= \frac{1}{\sigma_i^2}\sigma_i^2 \\ &= 1. \end{aligned}$$

Thus, the heteroskedasticity is corrected in the transformed model.

In matrix notation, the transformed model is:

$$\mathbf{wy} = \mathbf{wX} + \mathbf{w} ,$$

where $\mathbf{w}$ is the weight matrix used to standardize the variance. The weight matrix $\mathbf{w}$ is defined as:

$$\mathbf{w} = \begin{pmatrix} 1/\sigma_1 & 0 & 0 & \cdots & 0 \\ 0 & 1/\sigma_2 & 0 & \cdots & 0 \\ 0 & 0 & 1/\sigma_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1/\sigma_n \end{pmatrix}.$$

In the presence of heteroskedasticity, the variance of the error term, $Var(\epsilon_i|\mathbf{x}_i)$, is not constant across observations. This leads to inefficient OLS estimates.

Infeasible Weighted Least Squares (IWLS) assumes that the variances $\sigma_i^2 = Var(\epsilon_i|\mathbf{x}_i)$ are **known**. This allows us to adjust the regression equation to correct for heteroskedasticity.

The model is transformed as follows:

$$y_i = \mathbf{x}_i\beta + \epsilon_i \quad \text{(original equation)},$$

where ϵ_i has variance σ_i^2. To make the errors homoskedastic, we divide through by σ_i:

$$\frac{y_i}{\sigma_i} = \frac{\mathbf{x}_i}{\sigma_i}\beta + \frac{\epsilon_i}{\sigma_i}.$$

Now, the transformed error term ϵ_i/σ_i has a constant variance of 1:

$$Var\left(\frac{\epsilon_i}{\sigma_i}|\mathbf{x}_i\right) = 1.$$

The IWLS estimator minimizes the **weighted sum of squared residuals** for the transformed model:

$$\text{Minimize: } \sum_{i=1}^{n}\left(\frac{y_i - \mathbf{x}_i\beta}{\sigma_i}\right)^2.$$

In matrix form, the IWLS estimator is:

$$\hat{\beta}_{IWLS} = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{y},$$

where $\mathbf{W}$ is a diagonal matrix of weights:

$$\mathbf{W} = \begin{pmatrix} 1/\sigma_1^2 & 0 & \cdots & 0 \\ 0 & 1/\sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/\sigma_n^2 \end{pmatrix}.$$

Properties of IWLS

1. **Valid Standard Errors**:
   - If $Var(\epsilon_i|\mathbf{X}) = \sigma_i^2$, the usual standard errors from IWLS are valid.
2. **Robustness**:
   - If the variance assumption is incorrect ($Var(\epsilon_i|\mathbf{X}) \neq \sigma_i^2$), heteroskedasticity-robust standard errors must be used instead.

The primary issue with IWLS is that $\sigma_i^2 = Var(\epsilon_i|\mathbf{x}_i)$ is generally unknown. Specifically, we do not know:

$$\sigma_i^2 = Var(\epsilon_i|\mathbf{x}_i) = E(\epsilon_i^2|\mathbf{x}_i).$$

The challenges are:

1. **Single Observation**:
   - For each observation i, there is only one ϵ_i, which is insufficient to estimate the variance σ_i^2 directly.
2. **Dependence on Assumptions**:
   - To estimate σ_i^2, we must impose assumptions about its relationship to $\mathbf{x}_i$.

To make IWLS feasible, we model σ_i^2 as a function of the predictors $\mathbf{x}_i$. A common approach is:

$$\epsilon_i^2 = v_i \exp(\mathbf{x}_i \gamma),$$

where:

- v_i is an independent error term with strictly positive values, representing random noise.
- $\exp(\mathbf{x}_i \gamma)$ is a deterministic function of the predictors $\mathbf{x}_i$.

Taking the natural logarithm of both sides linearizes the model:

$$\ln(\epsilon_i^2) = \mathbf{x}_i \gamma + \ln(v_i),$$

where $\ln(v_i)$ is independent of $\mathbf{x}_i$. This transformation enables us to estimate γ using standard OLS techniques.

Estimation Procedure for Feasible GLS (FGLS)

Since we do not observe the true errors ϵ_i, we approximate them using the OLS residuals e_i. Here's the step-by-step process:

1. **Compute OLS Residuals**: First, fit the original model using OLS and calculate the residuals:

$$e_i = y_i - \mathbf{x}_i \hat{\beta}_{OLS}.$$

2. **Approximate ϵ_i^2 with e_i^2**: Use the squared residuals as a proxy for the squared errors:

$$e_i^2 \approx \epsilon_i^2.$$

3. **Log-Linear Model**: Fit the log-transformed model to estimate γ:

$$\ln(e_i^2) = \mathbf{x}_i \gamma + \ln(v_i).$$

Estimate γ using OLS, where $\ln(v_i)$ is treated as the error term.

4. **Estimate Variances**: Use the fitted values $\hat{\gamma}$ to estimate σ_i^2 for each observation:

$$\hat{\sigma}_i^2 = \exp(\mathbf{x}_i \hat{\gamma}).$$

5. **Perform Weighted Least Squares**: Use the estimated variances $\hat{\sigma}_i^2$ to construct the weight matrix $\hat{\mathbf{W}}$:

$$\hat{\mathbf{W}} = \begin{pmatrix} 1/\hat{\sigma}_1^2 & 0 & \cdots & 0 \\ 0 & 1/\hat{\sigma}_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/\hat{\sigma}_n^2 \end{pmatrix}.$$

Then, compute the **Feasible GLS (FGLS)** estimator:

$$\hat{\beta}_{FGLS} = (\mathbf{X}'\hat{\mathbf{W}}\mathbf{X})^{-1}\mathbf{X}'\hat{\mathbf{W}}\mathbf{y}.$$

5.2.2.2 Serial Correlation

Serial correlation (also called autocorrelation) occurs when the error terms in a regression model are correlated across observations. Formally:

$$Cov(\epsilon_i, \epsilon_j | \mathbf{X}) \neq 0 \quad \text{for } i \neq j.$$

This violates the Gauss-Markov assumption that $Cov(\epsilon_i, \epsilon_j | \mathbf{X}) = 0$, leading to inefficiencies in OLS estimates.

5.2.2.2.1 Covariance Stationarity If the errors are **covariance stationary**, the covariance between errors depends only on their relative time or positional difference (h), not their absolute position:

$$Cov(\epsilon_i, \epsilon_j | \mathbf{X}) = Cov(\epsilon_i, \epsilon_{i+h} | \mathbf{x}_i, \mathbf{x}_{i+h}) = \gamma_h,$$

where γ_h represents the covariance at lag h.

Under covariance stationarity, the variance-covariance matrix of the error term ϵ takes the following form:

$$Var(\epsilon|\mathbf{X}) = \Omega = \begin{pmatrix} \sigma^2 & \gamma_1 & \gamma_2 & \cdots & \gamma_{n-1} \\ \gamma_1 & \sigma^2 & \gamma_1 & \cdots & \gamma_{n-2} \\ \gamma_2 & \gamma_1 & \sigma^2 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \gamma_1 \\ \gamma_{n-1} & \gamma_{n-2} & \cdots & \gamma_1 & \sigma^2 \end{pmatrix}.$$

Key Points:

1. The diagonal elements represent the variance of the error term: σ^2.
2. The off-diagonal elements γ_h represent covariances at different lags h.

Why Serial Correlation Is a Problem?

The matrix Ω introduces n parameters to estimate (e.g., $\sigma^2, \gamma_1, \gamma_2, \dots, \gamma_{n-1}$). Estimating such a large number of parameters becomes impractical, especially for large datasets. To address this, we impose additional structure to reduce the number of parameters.

5.2.2.2.2 AR(1) Model In the **AR(1)** process, the errors follow a first-order autoregressive process:

$$\begin{aligned} y_t &= \beta_0 + x_t\beta_1 + \epsilon_t, \\ \epsilon_t &= \rho\epsilon_{t-1} + u_t, \end{aligned}$$

where:

- ρ is the first-order autocorrelation coefficient, capturing the relationship between consecutive errors.
- u_t is white noise, satisfying $Var(u_t) = \sigma_u^2$ and $Cov(u_t, u_{t-h}) = 0$ for $h \neq 0$.

Under the AR(1) assumption, the variance-covariance matrix of the error term ϵ becomes:

$$Var(\epsilon|\mathbf{X}) = \frac{\sigma_u^2}{1-\rho^2} \begin{pmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{n-1} \\ \rho & 1 & \rho & \cdots & \rho^{n-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{n-1} & \rho^{n-2} & \cdots & \rho & 1 \end{pmatrix}.$$

Key Features:

1. The diagonal elements represent the variance: $Var(\epsilon_t|\mathbf{X}) = \sigma_u^2/(1-\rho^2)$.
2. The off-diagonal elements decay exponentially with lag h: $Cov(\epsilon_t, \epsilon_{t-h}|\mathbf{X}) = \rho^h \cdot Var(\epsilon_t|\mathbf{X})$.

Under AR(1), only **one parameter** ρ needs to be estimated (in addition to σ_u^2), greatly simplifying the structure of Ω.

OLS Properties Under AR(1)

1. **Consistency**: If assumptions A1, A2, A3a, and A5a hold, OLS remains consistent.
2. **Asymptotic Normality**: OLS estimates are asymptotically normal.
3. **Inference with Serial Correlation**:
   - Standard OLS errors are invalid.
   - Use **Newey-West standard errors** to obtain robust inference.

5.2.2.2.3 Infeasible Cochrane-Orcutt The **Infeasible Cochrane-Orcutt** procedure addresses serial correlation in the error terms by assuming an **AR(1)** process for the errors:

$$\epsilon_t = \rho\epsilon_{t-1} + u_t,$$

where u_t is white noise and ρ is the autocorrelation coefficient.

By transforming the original regression equation:

$$y_t = \beta_0 + x_t\beta_1 + \epsilon_t,$$

we subtract ρ times the lagged equation:

$$\rho y_{t-1} = \rho(\beta_0 + x_{t-1}\beta_1 + \epsilon_{t-1}),$$

to obtain the weighted first-difference equation:

$$y_t - \rho y_{t-1} = (1-\rho)\beta_0 + (x_t - \rho x_{t-1})\beta_1 + u_t.$$

Key Points:

1. **Dependent Variable**: $y_t - \rho y_{t-1}$.
2. **Independent Variable**: $x_t - \rho x_{t-1}$.
3. **Error Term**: u_t, which satisfies the Gauss-Markov assumptions (A3, A4, A5).

The ICO estimator minimizes the sum of squared residuals for this transformed equation.

1. **Standard Errors**:

- If the errors truly follow an AR(1) process, the standard errors for the transformed equation are valid.
- For more complex error structures, **Newey-West HAC standard errors** are required.

2. **Loss of Observations**:
   - The transformation involves first differences, which means the first observation (y_1) cannot be used. This reduces the effective sample size by one.

The Problem: ρ Is Unknown

The ICO procedure is **infeasible** because it requires knowledge of ρ, the autocorrelation coefficient. In practice, we estimate ρ from the data.

To estimate ρ, we use the OLS residuals (e_t) as a proxy for the errors (ϵ_t). The estimate $\hat{\rho}$ is given by:

$$\hat{\rho} = \frac{\sum_{t=2}^{T} e_t e_{t-1}}{\sum_{t=2}^{T} e_t^2}.$$

Estimation via OLS:

1. Regress the OLS residuals e_t on their lagged values e_{t-1}, without an intercept:
$$e_t = \rho e_{t-1} + u_t.$$
2. The slope of this regression is the estimate $\hat{\rho}$.

This estimation is efficient under the AR(1) assumption and provides a practical approximation for ρ.

5.2.2.2.4 Feasible Prais-Winsten The **Feasible Prais-Winsten (FPW)** method addresses **AR(1) serial correlation** in regression models by transforming the data to eliminate serial dependence in the errors. Unlike the Infeasible Cochrane-Orcutt procedure, which discards the first observation, the Prais-Winsten method retains it using a weighted transformation.

The FPW transformation uses the following weighting matrix $\mathbf{w}$:

$$\mathbf{w} = \begin{pmatrix} \sqrt{1-\hat{\rho}^2} & 0 & 0 & \cdots & 0 \\ -\hat{\rho} & 1 & 0 & \cdots & 0 \\ 0 & -\hat{\rho} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & -\hat{\rho} & 1 \end{pmatrix}.$$

where

- The first row accounts for the transformation of the first observation, using $\sqrt{1-\hat{\rho}^2}$.
- Subsequent rows represent the AR(1) transformation for the remaining observations.

Step-by-Step Procedure

Step 1: Initial OLS Estimation

Estimate the regression model using OLS:

$$y_t = \mathbf{x}_t\beta + \epsilon_t,$$

and compute the residuals:

$$e_t = y_t - \mathbf{x}_t\hat{\beta}.$$

Step 2: Estimate the AR(1) Correlation Coefficient

Estimate the AR(1) correlation coefficient ρ by regressing e_t on e_{t-1} without an intercept:

$$e_t = \rho e_{t-1} + u_t.$$

The slope of this regression is the estimated $\hat{\rho}$.

Step 3: Transform the Data

Apply the transformation using the weighting matrix $\mathbf{w}$ to transform both the dependent variable $\mathbf{y}$ and the independent variables $\mathbf{X}$:

$$\mathbf{wy} = \mathbf{wX}\beta + \mathbf{w}\ .$$

Specifically: 1. For $t = 1$, the transformed dependent and independent variables are:

$$\tilde{y}_1 = \sqrt{1-\hat{\rho}^2} \cdot y_1, \quad \tilde{\mathbf{x}}_1 = \sqrt{1-\hat{\rho}^2} \cdot \mathbf{x}_1.$$

2. For $t = 2, \ldots, T$, the transformed variables are:

$$\tilde{y}_t = y_t - \hat{\rho} \cdot y_{t-1}, \quad \tilde{\mathbf{x}}_t = \mathbf{x}_t - \hat{\rho} \cdot \mathbf{x}_{t-1}.$$

Step 4: Feasible Prais-Winsten Estimation

Run OLS on the transformed equation:

$$\mathbf{wy} = \mathbf{wX}\beta + \mathbf{w} \ .$$

The resulting estimator is the **Feasible Prais-Winsten (FPW) estimator**:

$$\hat{\beta}_{FPW} = (\mathbf{X}'\mathbf{w}'\mathbf{wX})^{-1}\mathbf{X}'\mathbf{w}'\mathbf{wy}.$$

Properties of Feasible Prais-Winsten Estimator

1. **Infeasible Prais-Winsten Estimator**:
   - The infeasible Prais-Winsten (PW) estimator assumes the AR(1) parameter ρ is known.
   - Under assumptions A1, A2, and A3 for the unweighted equation, the infeasible PW estimator is **unbiased** and efficient.
2. **Feasible Prais-Winsten (FPW) Estimator**: The FPW estimator replaces the unknown ρ with an estimate $\hat{\rho}$ derived from the OLS residuals, introducing bias in small samples.
   1. **Bias**:
      - The FPW estimator is **biased** due to the estimation of $\hat{\rho}$, which introduces an additional layer of approximation.
   2. **Consistency**:
      1. The FPW estimator is **consistent** under the following assumptions:
         - A1: The model is linear in parameters.
         - A2: The independent variables are linearly independent.
         - A5: The data is generated through random sampling.
         - Additionally:

           $$E((\mathbf{x_t} - \mathbf{x_{t-1}})'(\epsilon_t - \rho\epsilon_{t-1})) = 0.$$

           This condition ensures the transformed error term $\epsilon_t - \rho\epsilon_{t-1}$ is uncorrelated with the transformed regressors $\mathbf{x_t} - \mathbf{x_{t-1}}$.
      2. **Note**: A3a (zero conditional mean of the error term, $E(\epsilon_t|\mathbf{x}_t) = 0$) is **not sufficient** for the above condition. Full exogeneity of the independent variables (A3) is required.
   3. **Efficiency**
      1. **Asymptotic Efficiency**: The FPW estimator is asymptotically **more efficient than OLS** if the errors are truly generated by an AR(1) process:

         $$\epsilon_t = \rho\epsilon_{t-1} + u_t, \quad Var(u_t) = \sigma^2.$$

      2. **Standard Errors**:

1. **Usual Standard Errors**: If the errors are correctly specified as an AR(1) process, the usual standard errors from FPW are valid.
2. **Robust Standard Errors**: If there is concern about a more complex dependence structure (e.g., higher-order autocorrelation or heteroskedasticity), use **Newey-West Standard Errors** for inference. These are robust to both serial correlation and heteroskedasticity.

5.2.2.3 Cluster Errors

Consider the regression model with clustered errors:

$$y_{gi} = \mathbf{x}_{gi}\beta + \epsilon_{gi},$$

where:

- g indexes the group (e.g., households, firms, schools).
- i indexes the individual within the group.

The covariance structure for the errors ϵ_{gi} is defined as:

$$Cov(\epsilon_{gi}, \epsilon_{hj}) \begin{cases} = 0 & \text{if } g \neq h \text{ (independent across groups)}, \\ \neq 0 & \text{for any pair } (i,j) \text{ within group } g. \end{cases}$$

Within each group, individuals' errors may be correlated (i.e., intra-group correlation), while errors are independent across groups. This violates A4 (constant variance and no correlation of errors).

Suppose there are three groups with varying sizes. The variance-covariance matrix Ω for the errors ϵ is:

$$Var(\epsilon|\mathbf{X}) = \Omega = \begin{pmatrix} \sigma^2 & \delta^1_{12} & \delta^1_{13} & 0 & 0 & 0 \\ \delta^1_{12} & \sigma^2 & \delta^1_{23} & 0 & 0 & 0 \\ \delta^1_{13} & \delta^1_{23} & \sigma^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2 & \delta^2_{12} & 0 \\ 0 & 0 & 0 & \delta^2_{12} & \sigma^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2 \end{pmatrix}.$$

where

- $\delta^g_{ij} = Cov(\epsilon_{gi}, \epsilon_{gj})$ is the covariance between errors for individuals i and j in group g.

- $Cov(\epsilon_{gi}, \epsilon_{hj}) = 0$ for $g \neq h$ (independent groups).

Infeasible Generalized Least Squares (Cluster)

1. **Assume Known Variance-Covariance Matrix**: If σ^2 and δ_{ij}^g are known, construct Ω and compute its inverse Ω^{-1}.

2. **Infeasible GLS Estimator**: The infeasible generalized least squares (IGLS) estimator is:

$$\hat{\beta}_{IGLS} = (\mathbf{X}'\Omega^{-1}\mathbf{X})^{-1}\mathbf{X}'\Omega^{-1}\mathbf{y}.$$

Problem:

- We do not know σ^2 and δ_{ij}^g, making this approach **infeasible**.
- Even if Ω is estimated, incorrect assumptions about its structure may lead to invalid inference.

To make the estimation feasible, we assume a **group-level random effects** specification for the error:

$$\begin{aligned} y_{gi} &= \mathbf{x}_{gi}\beta + c_g + u_{gi}, \\ Var(c_g|\mathbf{x}_i) &= \sigma_c^2, \\ Var(u_{gi}|\mathbf{x}_i) &= \sigma_u^2, \end{aligned}$$

where:

- c_g represents the **group-level random effect** (common shocks within each group, independent across groups).

- u_{gi} represents the **individual-level error** (idiosyncratic shocks within each group, independent across individuals and groups).

- $\epsilon_{gi} = c_g + u_{gi}$

Independence Assumptions:

- c_g and u_{gi} are independent of each other.
- Both are mean-independent of $\mathbf{x}_i$.

Under this specification, the variance-covariance matrix Ω becomes block diagonal, where each block corresponds to a group:

$$Var(\epsilon|\mathbf{X}) = \Omega = \begin{pmatrix} \sigma_c^2 + \sigma_u^2 & \sigma_c^2 & \sigma_c^2 & 0 & 0 & 0 \\ \sigma_c^2 & \sigma_c^2 + \sigma_u^2 & \sigma_c^2 & 0 & 0 & 0 \\ \sigma_c^2 & \sigma_c^2 & \sigma_c^2 + \sigma_u^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_c^2 + \sigma_u^2 & \sigma_c^2 & 0 \\ 0 & 0 & 0 & \sigma_c^2 & \sigma_c^2 + \sigma_u^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_c^2 + \sigma_u^2 \end{pmatrix}.$$

When the variance components σ_c^2 and σ_u^2 are unknown, we can use the **Feasible Group-Level Random Effects (RE)** estimator to simultaneously estimate these variances and the regression coefficients β. This practical approach allows us to account for intra-group correlation in the errors and still obtain consistent and efficient estimates of the parameters.

Step-by-Step Procedure

Step 1: Initial OLS Estimation

Estimate the regression model using OLS:

$$y_{gi} = \mathbf{x}_{gi}\beta + \epsilon_{gi},$$

and compute the residuals:

$$e_{gi} = y_{gi} - \mathbf{x}_{gi}\widehat{\beta}.$$

Step 2: Estimate Variance Components

Use the standard OLS variance estimator s^2 to estimate the **total variance**:

$$s^2 = \frac{1}{n-k}\sum_{i=1}^{n} e_i^2,$$

where n is the total number of observations and k is the number of regressors (including the intercept).

Estimate the **between-group variance** $\widehat{\sigma}_c^2$ using:

$$\widehat{\sigma}_c^2 = \frac{1}{G}\sum_{g=1}^{G}\left(\frac{1}{\sum_{i=1}^{n_g-1} i}\sum_{i\neq j}\sum_{j=1}^{n_g} e_{gi}e_{gj}\right),$$

where:

- G is the total number of groups,
- n_g is the size of group g,
- The term $\sum_{i \neq j} e_{gi} e_{gj}$ accounts for within-group covariance.

Estimate the **within-group variance** as:

$$\hat{\sigma}_u^2 = s^2 - \hat{\sigma}_c^2.$$

Step 3: Construct the Variance-Covariance Matrix

Use the estimated variances $\hat{\sigma}_c^2$ and $\hat{\sigma}_u^2$ to construct the variance-covariance matrix $\hat{\Omega}$ for the error term:

$$\hat{\Omega}_{gi,gj} = \begin{cases} \hat{\sigma}_c^2 + \hat{\sigma}_u^2 & \text{if } i = j \text{ (diagonal elements)}, \\ \hat{\sigma}_c^2 & \text{if } i \neq j \text{ (off-diagonal elements within group)}, \\ 0 & \text{if } g \neq h \text{ (across groups)}. \end{cases}$$

Step 4: Feasible GLS Estimation

With $\hat{\Omega}$ in hand, perform Feasible Generalized Least Squares (FGLS) to estimate β:

$$\hat{\beta}_{RE} = (\mathbf{X}'\hat{\Omega}^{-1}\mathbf{X})^{-1}\mathbf{X}'\hat{\Omega}^{-1}\mathbf{y}.$$

If the assumptions about Ω are incorrect or infeasible, use **cluster-robust standard errors** to account for intra-group correlation without explicitly modeling the variance-covariance structure. These standard errors remain valid under arbitrary within-cluster dependence, provided clusters are independent.

Properties of the Feasible Group-Level Random Effects Estimator

1. **Infeasible Group RE Estimator**

- The infeasible RE estimator (assuming known variances) is **unbiased** under assumptions A1, A2, and A3 for the unweighted equation.
- A3 requires:
$$E(\epsilon_{gi}|\mathbf{x}_i) = E(c_g|\mathbf{x}_i) + E(u_{gi}|\mathbf{x}_i) = 0.$$
This assumes:
  - $E(c_g|\mathbf{x}_i) = 0$: The **random effects assumption** (group-level effects are uncorrelated with the regressors).
  - $E(u_{gi}|\mathbf{x}_i) = 0$: No endogeneity at the individual level.

2. **Feasible Group RE Estimator**

- The feasible RE estimator is **biased** because the variances σ_c^2 and σ_u^2 are estimated, introducing approximation errors.
- However, the estimator is **consistent** under A1, A2, A3a ($E(\mathbf{x}_i'\epsilon_{gi}) = E(\mathbf{x}_i'c_g) + E(\mathbf{x}_i'u_{gi}) = 0$), A5a.
- **Efficiency**
  - **Asymptotic Efficiency**:
    - The feasible RE estimator is asymptotically more efficient than OLS if the errors follow the random effects specification.
  - **Standard Errors**:
    - If the random effects specification is correct, the usual standard errors are consistent.
    - If there is concern about more complex dependence structures or heteroskedasticity, use **cluster robust standard errors**.

5.2.3 Weighted Least Squares

In the presence of heteroskedasticity, the errors ϵ_i have non-constant variance $Var(\epsilon_i|\mathbf{x}_i) = \sigma_i^2$. This violates the Gauss-Markov assumption of homoskedasticity, leading to inefficient OLS estimates.

Weighted Least Squares (WLS) addresses this by applying weights inversely proportional to the variance of the errors, ensuring that observations with larger variances have less influence on the estimation.

- Weighted Least Squares is essentially Generalized Least Squares in the special case that is a *diagonal* matrix with variances σ_i^2 on the diagonal (i.e., errors are uncorrelated but have non-constant variance).
  - That is, assume the errors are uncorrelated but **heteroskedastic**: $= \text{diag}(\sigma_1^2, \dots, \sigma_n^2)$
  - Then $^{-1} = \text{diag}(1/\sigma_1^2, \dots, 1/\sigma_n^2)$

Steps for Feasible Weighted Least Squares (FWLS)

1. Initial OLS Estimation

First, estimate the model using OLS:

$$y_i = \mathbf{x}_i\beta + \epsilon_i,$$

and compute the residuals:

$$e_i = y_i - \mathbf{x}_i\widehat{\beta}.$$

2. Model the Error Variance

Transform the residuals to model the variance as a function of the predictors:

$$\ln(e_i^2) = \mathbf{x}_i\gamma + \ln(v_i),$$

where:

- e_i^2 approximates ϵ_i^2,
- $\ln(v_i)$ is the error term in this auxiliary regression, assumed independent of $\mathbf{x}_i$.

Estimate this equation using OLS to obtain the predicted values:

$$\hat{g}_i = \mathbf{x}_i\hat{\gamma}.$$

3. Estimate Weights

Use the predicted values from the auxiliary regression to compute the weights:

$$\hat{\sigma}_i = \sqrt{\exp(\hat{g}_i)}.$$

These weights approximate the standard deviation of the errors.

4. Weighted Regression

Transform the original equation by dividing through by $\hat{\sigma}_i$:

$$\frac{y_i}{\hat{\sigma}_i} = \frac{\mathbf{x}_i}{\hat{\sigma}_i}\beta + \frac{\epsilon_i}{\hat{\sigma}_i}.$$

Estimate the transformed equation using OLS. The resulting estimator is the **Feasible Weighted Least Squares (FWLS)** estimator:

$$\hat{\beta}_{FWLS} = (\mathbf{X}'\hat{\mathbf{W}}\mathbf{X})^{-1}\mathbf{X}'\hat{\mathbf{W}}\mathbf{y},$$

where $\hat{\mathbf{W}}$ is a diagonal weight matrix with elements $1/\hat{\sigma}_i^2$.

Properties of the FWLS Estimator

1. **Unbiasedness**:
   - The infeasible WLS estimator (where σ_i is known) is unbiased under assumptions A1-A3 for the unweighted model.
   - The **FWLS estimator** is **not unbiased** due to the approximation of σ_i using $\hat{\sigma}_i$.
2. **Consistency**:
   - The FWLS estimator is **consistent** under the following assumptions:

  - A1 (for the unweighted equation): The model is linear in parameters.
  - A2 (for the unweighted equation): The independent variables are linearly independent.
  - A5: The data is randomly sampled.
  - $E(\mathbf{x}_i'\epsilon_i/\sigma_i^2) = 0$. A3a: Weaker Exogeneity Assumption is not sufficient, but A3 is.
3. **Efficiency**:
   - The FWLS estimator is asymptotically more efficient than OLS if the errors have multiplicative exponential heteroskedasticity:

$$Var(\epsilon_i|\mathbf{x}_i) = \sigma_i^2 = \exp(\mathbf{x}_i\gamma).$$

The FWLS estimator is asymptotically more efficient than OLS if the errors have multiplicative exponential heteroskedasticity.

1. **Usual Standard Errors**:
   - If the errors are truly multiplicative exponential heteroskedastic, the usual standard errors for FWLS are valid.
2. **Heteroskedastic Robust Standard Errors**:
   - If there is potential mis-specification of the multiplicative exponential model for σ_i^2, heteroskedastic-robust standard errors should be reported to ensure valid inference.

5.3 Maximum Likelihood

The **Maximum Likelihood Estimation (MLE)** is a statistical method used to estimate the parameters of a model by maximizing the likelihood of observing the given data. The premise is to find the parameter values that maximize the probability (or likelihood) of the observed data.

The likelihood function, denoted as $L(\theta)$, is expressed as:

$$L(\theta) = \prod_{i=1}^{n} f(y_i|\theta)$$

where:

- $f(y|\theta)$ is the probability density or mass function of observing a single value of Y given the parameter θ.
- The product runs over all n observations.

For different types of data, $f(y|\theta)$ can take different forms. For example, if y is dichotomous (e.g., success/failure), then the likelihood function becomes:

$$L(\theta) = \prod_{i=1}^{n} \theta^{y_i}(1-\theta)^{1-y_i}$$

Here, $\hat{\theta}$ is the Maximum Likelihood Estimator (MLE) if:

$$L(\hat{\theta}) > L(\theta_0), \quad \forall \theta_0 \text{ in the parameter space.}$$

See Distributions for a review on variable distributions.

5.3.1 Motivation for MLE

Suppose we know the **conditional distribution** of Y given X, denoted as:

$$f_{Y|X}(y, x; \theta)$$

where θ is an unknown parameter of the distribution. Sometimes, we are only concerned with the unconditional distribution $f_Y(y; \theta)$.

For a sample of independent and identically distributed (i.i.d.) data, the joint probability of the sample is:

$$f_{Y_1,\dots,Y_n|X_1,\dots,X_n}(y_1, \dots, y_n, x_1, \dots, x_n; \theta) = \prod_{i=1}^{n} f_{Y|X}(y_i, x_i; \theta)$$

The **joint distribution**, evaluated at the observed data, defines the likelihood function. The goal of MLE is to find the parameter θ that maximizes this likelihood.

To estimate θ, we maximize the likelihood function:

$$\max_{\theta} \prod_{i=1}^{n} f_{Y|X}(y_i, x_i; \theta)$$

In practice, it is easier to work with the natural logarithm of the likelihood (log-likelihood), as it transforms the product into a sum:

$$\max_{\theta} \sum_{i=1}^{n} \ln(f_{Y|X}(y_i, x_i; \theta))$$

Solving for the Maximum Likelihood Estimator

1. **First-Order Condition**: Solve the first derivative of the log-likelihood function with respect to θ:

$$\frac{\partial}{\partial \theta} \ell(\theta) \;=\; \frac{\partial}{\partial \theta} \ln L(\theta) \;=\; \frac{\partial}{\partial \theta} \sum_{i=1}^{n} \ln(f_{Y|X}(y_i, x_i; \hat{\theta}_{MLE})) = 0$$

This yields the critical points where the likelihood is maximized. This derivative, sometimes written as $U(\theta)$, is called the **score**. Intuitively, the log-likelihood's "peak" indicates the parameter value(s) that make the observed data "most likely."

2. **Second-Order Condition**: Verify that the second derivative of the log-likelihood function is negative at the critical point:

$$\frac{\partial^2}{\partial\theta^2}\sum_{i=1}^{n}\ln(f_{Y|X}(y_i, x_i; \hat{\theta}_{MLE})) < 0$$

This ensures that the solution corresponds to a maximum.

Examples of Likelihood Functions

1. Unconditional Poisson Distribution

The Poisson distribution models count data, such as the number of website visits in a day or product orders per hour. Its likelihood function is:

$$L(\theta) = \prod_{i=1}^{n}\frac{\theta^{y_i}e^{-\theta}}{y_i!}$$

2. Exponential Distribution

The exponential distribution is often used to model the time between events, such as the time until a machine fails. Its probability density function (PDF) is:

$$f_{Y|X}(y, x; \theta) = \frac{\exp(-y/(x\theta))}{x\theta}$$

The joint likelihood for n observations is:

$$L(\theta) = \prod_{i=1}^{n}\frac{\exp(-y_i/(x_i\theta))}{x_i\theta}$$

By taking the logarithm, we obtain the log-likelihood for ease of maximization.

5.3.2 Key Quantities for Inference

1. **Score Function**
   The **score** is given by

$$U(\theta) \;=\; \frac{d}{d\theta}\ell(\theta).$$

Setting $U(\hat{\theta}_{\text{MLE}}) = 0$ yields the critical points of the log-likelihood, from which we can find $\hat{\theta}_{\text{MLE}}$.

2. **Observed Information**
   The second derivative of the log-likelihood, taken at the MLE, is called the **observed information**:

$$I_O(\theta) \;=\; -\frac{d^2}{d\theta^2}\ell(\theta).$$

   (The negative sign is often included so that $I_O(\theta)$ is *positive* if $\ell(\theta)$ is concave near its maximum. In some texts, you will see it defined without the negative sign, but the idea is the same: it measures the "pointedness" or curvature of $\ell(\theta)$ at its maximum.)

3. **Fisher Information**
   The **Fisher Information** (or **expected information**) is the expectation of the observed information over the distribution of the data:

$$I(\theta) \;=\; \mathbb{E}\big[I_O(\theta)\big].$$

   It quantifies how much information the data carry about the parameter θ. A larger Fisher information suggests that you can estimate θ more precisely.

4. **Approximate Variance of $\hat{\theta}_{\text{MLE}}$**
   One of the key results from standard asymptotic theory is that, for large n, the variance of $\hat{\theta}_{\text{MLE}}$ can be approximated by the inverse of the Fisher information:

$$\text{Var}(\hat{\theta}_{\text{MLE}}) \;\approx\; I(\theta)^{-1}.$$

   This also lays the groundwork for constructing confidence intervals for θ in large samples.

5.3.3 Assumptions of MLE

MLE has desirable properties—*consistency*, *asymptotic normality*, and *efficiency*—but these do not come "for free." Instead, they rely on certain assumptions. Below is a breakdown of the main **regularity conditions**. These conditions are typically mild in many practical settings (for example, in exponential families, such as the normal distribution), but need to be checked in more complex models.

High-Level Regulatory Assumptions

1. **Independence and Identical Distribution (iid)**
   The sample $\{(x_i, y_i)\}$ is usually assumed to be composed of independent and identically distributed observations. This independence assumption simplifies the likelihood to a product of individual densities:

   $$L(\theta) = \prod_{i=1}^{n} f_{Y|X}(y_i, x_i; \theta).$$

   In practice, if you have dependent data (e.g., time series, spatial data), modifications are required in the likelihood function.

2. **Same Density Function**
   All observations must come from the *same* conditional probability density function $f_{Y|X}(\cdot, \cdot; \theta)$. If the model changes across observations, you cannot simply multiply all of them together in one unified likelihood.

3. **Multivariate Normality (for certain models)**
   In many practical cases—especially for continuous outcomes—you might assume (multivariate) normal distributions with finite second or fourth moments (Little, 1988). Under these assumptions, the MLE for the mean vector and covariance matrix is consistent and (under further conditions) asymptotically normal. This assumption is quite common in regression, ANOVA, and other classical statistical frameworks.

5.3.3.1 Large Sample Properties of MLE

5.3.3.1.1 Consistency of MLE **Definition:** An estimator $\hat{\theta}_n$ is *consistent* if it converges in probability to the true parameter value θ_0 as the sample size $n \to \infty$:

$$\hat{\theta}_n \ \to^p \ \theta_0.$$

For the MLE, a set of regularity conditions $R1$–$R4$ is commonly used to ensure consistency:

1. **R1**
   If $\theta \neq \theta_0$, then

   $$f_{Y|X}(y_i, x_i; \theta) \ \neq \ f_{Y|X}(y_i, x_i; \theta_0).$$

   In simpler terms, the model is identifiable: no two distinct parameter values generate the *exact* same distribution for the data.

2. **R2**
   The parameter space Θ is compact (closed and bounded), and it contains

the true parameter θ_0. This ensures that θ lies in a "nice" region (no parameter going to infinity, etc.), making it easier to prove that a maximum in that space indeed exists.

3. **R3**
   The log-likelihood function $\ln(f_{Y|X}(y_i, x_i; \theta))$ is continuous in θ with probability 1. Continuity is important so that we can apply theorems (like the Continuous Mapping Theorem or the Extreme Value Theorem) to find maxima.

4. **R4**
   The expected supremum of the absolute value of the log-likelihood is finite:

$$\mathbb{E}\Big(\sup_{\theta\in\Theta}\big|\ln(f_{Y|X}(y_i, x_i; \theta))\big|\Big) \;<\; \infty.$$

   This is a technical condition that helps ensure we can "exchange" expectations and suprema, a step needed in many consistency proofs.

When these conditions are satisfied, you can show via standard arguments (e.g., the Law of Large Numbers, uniform convergence of the log-likelihood) that:

$$\hat{\theta}_{\text{MLE}} \;\to^p\; \theta_0 \quad \text{(consistency)}.$$

5.3.3.1.2 Asymptotic Normality of MLE **Definition:** An estimator $\hat{\theta}_n$ is *asymptotically normal* if

$$\sqrt{n}\,(\hat{\theta}_n - \theta_0) \;\to^d\; \mathcal{N}(0, \Sigma),$$

where $\to^d$ denotes convergence in distribution and Σ is some covariance matrix. For the MLE, Σ is typically $I(\theta_0)^{-1}$, where $I(\theta_0)$ is the Fisher information evaluated at the true parameter.

Beyond $R1$–$R4$, we need the following additional assumptions:

1. **R5**
   The true parameter θ_0 is in the *interior* of the parameter space Θ. If θ_0 sits on the boundary, different arguments are required to handle edge effects.

2. **R6**
   The pdf $f_{Y|X}(y_i, x_i; \theta)$ is *twice* continuously differentiable (in θ) and strictly positive in a neighborhood N of θ_0. This allows us to use second-order Taylor expansions around θ_0 to get the approximate distribution of $\hat{\theta}_{\text{MLE}}$.

3. **R7**
   The following integrals are finite in some neighborhood N of θ_0:
   - $\int \sup_{\theta \in N} \left\| \partial f_{Y|X}(y_i, x_i; \theta)/\partial\theta \right\| d(y, x) < \infty.$
   - $\int \sup_{\theta \in N} \left\| \partial^2 f_{Y|X}(y_i, x_i; \theta)/\partial\theta\partial\theta' \right\| d(y, x) < \infty.$
   - $\mathbb{E}\Big(\sup_{\theta \in N} \left\| \partial^2 \ln(f_{Y|X}(y_i, x_i; \theta))/\partial\theta\partial\theta' \right\| \Big) < \infty.$

   These conditions ensure that differentiating inside integrals is justified (via the dominated convergence theorem) and that we can expand the log-likelihood in a Taylor series safely.

4. **R8**
   The information matrix $I(\theta_0)$ exists and is nonsingular:

$$I(\theta_0) \;=\; \mathrm{Var}\Big(\frac{\partial}{\partial\theta} \ln(f_{Y|X}(y_i, x_i; \theta_0)) \Big) \;\neq\; 0.$$

   Nonsingularity implies there is enough information in the data to estimate θ uniquely.

Under $R1$–$R8$, you can show that

$$\sqrt{n}\,(\hat{\theta}_{\mathrm{MLE}} - \theta_0) \;\to^d\; \mathcal{N}\Big(0,\, I(\theta_0)^{-1}\Big).$$

This result is central to frequentist inference, allowing you to construct approximate confidence intervals and hypothesis tests using the normal approximation for large n.

5.3.4 Properties of MLE

Having established in earlier sections that Maximum Likelihood Estimators (MLEs) are **consistent** (Consistency of MLE) and **asymptotically normal** (Asymptotic Normality of MLE) under standard regularity conditions, we now highlight additional properties that make MLE a powerful estimation technique.

1. **Asymptotic Efficiency**

- **Definition**: An estimator is *asymptotically efficient* if it attains the smallest possible asymptotic variance among all consistent estimators (i.e., it achieves the *Cramér-Rao Lower Bound*).
- **Interpretation**: In large samples, MLE typically has smaller standard errors than other consistent estimators that do not fully use the assumed distributional form.
- **Implication**: When the true model is correctly specified, MLE is the *most efficient* among a broad class of estimators, leading to more precise inference for θ.

  - **Cramér-Rao Lower Bound (CRLB)**: A theoretical lower limit on the variance of any unbiased (or asymptotically unbiased) estimator (Cramér, 1999, Rao (1992)).
  - **When MLE Meets CRLB**: Under correct specification and standard regularity conditions, the asymptotic variance of the MLE matches the CRLB, making it *asymptotically efficient*.
  - **Interpretation**: Achieving CRLB means no other unbiased estimator can consistently outperform MLE in terms of variance for large n.

2. **Invariance**

- **Core Idea**: If $\hat{\theta}$ is the MLE for θ, then for any *smooth* transformation $g(\theta)$, the MLE for $g(\theta)$ is simply $g(\hat{\theta})$.
- **Example**: If θ is a mean parameter and you want the MLE for the *variance* θ^2, you can just square the MLE for θ.
- **Key Point**: This *invariance property* saves considerable effort—there is no need to re-derive a new likelihood for the transformed parameter.

3. Explicit vs. Implicit MLE

- **Explicit MLE**:
  Occurs when the score equation can be solved in *closed form*. A classic example is the MLE for the mean and variance in a normal distribution.
- **Implicit MLE**:
  Happens when no closed-form solution exists. Iterative numerical methods, such as **Newton-Raphson**, **Expectation-Maximization (EM)**, or other optimization algorithms, are used to find $\hat{\theta}$.

Distributional Mis-Specification

- **Definition**: If you assume a distribution for $f_{Y|X}(\cdot;\theta)$ that does *not* reflect the true data-generating process, the MLE may become *inconsistent* or biased in finite samples.
- **Quasi-MLE**:
  - A strategy to handle certain forms of mis-specification.
  - If the chosen distribution belongs to a flexible class or meets certain conditions (e.g., generalized linear models with a robust link), the resulting parameter estimates can remain consistent *for some parameters of interest*.
- **Nonparametric & Semiparametric Approaches**:
  - Require minimal or no distributional assumptions.
  - More robust to mis-specification but can be *harder to implement* and may exhibit higher variance or require larger sample sizes to achieve comparable precision.

5.3.5 Practical Considerations

1. **Use Cases**
   - MLE is extremely popular for:
     - **Binary Outcomes** (logistic regression)
     - **Count Data** (Poisson regression)
     - **Strictly Positive Outcomes** (Gamma regression)
     - **Heteroskedastic Settings** (models with variance related to mean, e.g., GLMs)
2. **Distributional Assumptions**
   - The efficiency gains of MLE stem from using a specific probability model.
   - If the assumed model closely reflects the data-generating process, MLE gives accurate parameter estimates and reliable standard errors.
   - MLE assumes knowledge of the conditional distribution of the outcome variable. This assumption parallels the normality assumption in linear regression models (e.g., A6 Normal Distribution).
   - If severely mis-specified, consider robust or semi-/nonparametric methods.
3. **Comparison with OLS**: See Comparison of MLE and OLS for more details.
   - Ordinary Least Squares is a special case of MLE when errors are normally distributed and homoscedastic.
   - In more general settings (e.g., non-Gaussian or heteroskedastic data), MLE can outperform OLS in terms of smaller standard errors and better inference.
4. **Numerical Stability & Computation**
   - For complex likelihoods, iterative methods can fail to converge or converge to local maxima.
   - Proper initialization and diagnostics (e.g., checking multiple start points) are crucial.

5.3.6 Comparison of MLE and OLS

While Maximum Likelihood Estimation is a powerful estimation method, it does not solve all of the challenges associated with Ordinary Least Squares. Below is a detailed comparison highlighting similarities, differences, and limitations.

Key Points of Comparison

1. **Inference Methods**:
   - **MLE**:
     - Joint inference is typically conducted using **log-likelihood calculations**, such as likelihood ratio tests or information criteria (e.g., AIC, BIC).
     - These methods replace the use of F-statistics commonly associated with OLS.

- **OLS**:
  - Relies on the **F-statistic** for hypothesis testing and joint inference.

2. **Sensitivity to Functional Form**:
   - Both MLE and OLS are sensitive to the **functional form** of the model. Incorrect specification (e.g., linear vs. nonlinear relationships) can lead to biased or inefficient estimates in both cases.
3. **Perfect Collinearity and Multicollinearity**:
   - Both methods are affected by collinearity:
     - **Perfect collinearity** (e.g., two identical predictors) makes parameter estimation impossible.
     - **Multicollinearity** (highly correlated predictors) inflates standard errors, reducing the precision of estimates.
   - Neither MLE nor OLS directly resolves these issues without additional measures, such as regularization or variable selection.
4. **Endogeneity**:
   - Problems like **omitted variable bias** or **simultaneous equations** affect both MLE and OLS:
     - If relevant predictors are omitted, estimates from both methods are likely to be biased and inconsistent.
     - Similarly, in systems of simultaneous equations, both methods yield biased results unless endogeneity is addressed through instrumental variables or other approaches.
   - **MLE**, while efficient under correct model specification, does not inherently address endogeneity.

Situations Where MLE and OLS Differ

Aspect	MLE	OLS
Estimator Efficiency	Efficient for correctly specified distributions.	Efficient under Gauss-Markov assumptions.
Assumptions about Errors	Requires specifying a distribution (e.g., normal, binomial).	Requires only mean-zero errors and homoscedasticity.
Use of Likelihood	Based on maximizing the likelihood function for parameter estimation.	Based on minimizing the sum of squared residuals.
Model Flexibility	More flexible (supports various distributions, non-linear models).	Primarily linear models (extensions for non-linear exist).
Interpretation	Log-likelihood values guide model comparison (AIC/BIC).	R-squared and adjusted R-squared measure fit.

Practical Considerations

1. **When to Use MLE**:
   - Situations where the dependent variable is:
     - Binary (e.g., logistic regression)
     - Count data (e.g., Poisson regression)
     - Skewed or bounded (e.g., survival models)
   - When the model naturally arises from a probabilistic framework.
2. **When to Use OLS**:
   - Suitable for continuous dependent variables with approximately linear relationships between predictors and outcomes.
   - Simpler to implement and interpret when the assumptions of linear regression are reasonably met.

5.3.7 Applications of MLE

MLE is widely used across various applications to estimate parameters in models tailored for specific data structures. Below are key applications of MLE, categorized by problem type and estimation method.

Model Type	Examples	Key Characteristics	Common Estimation Methods	Additional Notes
Corner Solution Models	Hours worked Donations to charity Household consumption of a good	Dependent variable is often **censored at zero** (or another threshold). Large fraction of observations at the corner (e.g., 0 hours, 0 donations).	Tobit regression (latent variable approach with censoring)	Useful when a continuous outcome has a **mass point** at zero but also positive values (e.g., 30% of individuals donate \$0, the rest donate > \$0).

Model Type	Examples	Key Characteristics	Common Estimation Methods	Additional Notes
Non-Negative Count Models	Number of arrests Number of cigarettes smoked Doctor visits per year	Dependent variable consists of **non-negative integer counts**. Possible **overdispersion** (variance > mean).	Poisson regression, Negative Binomial regression	Poisson assumes mean = variance, so often Negative Binomial is preferred for real data. Zero-inflated models (ZIP/ZINB) may be used for data with **excess zeros**.
Multinomial Choice Models	Demand for different car brands Votes in a primary election Choice of travel mode	Dependent variable is a **categorical choice** among **3+ alternatives**. Each category is distinct, with no inherent ordering (e.g., brand A, B, or C).	Multinomial logit, Multinomial probit	Extension of binary choice (logit/probit) to multiple categories. **Independence of Irrelevant Alternatives (IIA)** can be a concern for the multinomial logit.
Ordinal Choice Models	Self-reported happiness (low/medium/high) Income level brackets Likert-scale surveys	Dependent variable is **ordered** (e.g., **low < medium < high**). Distances between categories are not necessarily equal.	Ordered logit, Ordered probit	Probit/logit framework adapted to preserve **ordinal information**. Interprets latent continuous variable mapped to discrete ordered categories.

5.3.7.1 Binary Response Models

A binary response variable (y_i) follows a Bernoulli distribution:

$$f_Y(y_i; p) = p^{y_i}(1-p)^{(1-y_i)}$$

where p is the probability of success. For conditional models, the likelihood becomes:

$$f_{Y|X}(y_i, x_i; p(.)) = p(x_i)^{y_i}(1-p(x_i))^{(1-y_i)}$$

To model $p(x_i)$, we use a function of x_i and unknown parameters θ. A common approach involves a **latent variable model**:

$$\begin{aligned} y_i &= 1\{y_i^* > 0\}, \\ y_i^* &= x_i\beta - \epsilon_i, \end{aligned}$$

where:

- y_i^* is an unobserved (latent) variable.
- ϵ_i is a random variable with mean 0, representing unobserved noise.

Rewriting in terms of observed data:

$$y_i = 1\{x_i\beta > \epsilon_i\}.$$

The probability function becomes:

$$\begin{aligned} p(x_i) &= P(y_i = 1|x_i) \\ &= P(x_i\beta > \epsilon_i|x_i) \\ &= F_{\epsilon|X}(x_i\beta|x_i), \end{aligned}$$

where $F_{\epsilon|X}(.)$ is the cumulative distribution function (CDF) of ϵ_i. Assuming independence of ϵ_i and x_i, the probability function simplifies to:

$$p(x_i) = F_\epsilon(x_i\beta).$$

The conditional expectation function is equivalent:

$$E(y_i|x_i) = P(y_i = 1|x_i) = F_\epsilon(x_i\beta).$$

Common Distributional Assumptions

1. **Probit Model**:
   - Assumes ϵ_i follows a standard normal distribution.
   - $F_\epsilon(.) = \Phi(.)$, where $\Phi(.)$ is the standard normal CDF.
2. **Logit Model**:

- Assumes ϵ_i follows a standard logistic distribution.
- $F_\epsilon(.) = \Lambda(.)$, where $\Lambda(.)$ is the logistic CDF.

Steps to Derive MLE for Binary Models

1. **Specify the Log-Likelihood**:
   - For a chosen distribution (e.g., normal for Probit or logistic for Logit), the log-likelihood is:

$$\ln(f_{Y|X}(y_i, x_i; \beta)) = y_i \ln(F_\epsilon(x_i\beta)) + (1 - y_i)\ln(1 - F_\epsilon(x_i\beta)).$$

2. **Maximize the Log-Likelihood**:
   - Find the parameter estimates that maximize the log-likelihood:

$$\hat{\beta}_{MLE} = \underset{\beta}{\text{argmax}} \sum_{i=1}^{n} \ln(f_{Y|X}(y_i, x_i; \beta)).$$

Properties of Probit and Logit Estimators

- **Consistency and Asymptotic Normality**:
  - Probit and Logit estimators are consistent and asymptotically normal if:
    - A2 Full Rank: $E(x_i'x_i)$ exists and is non-singular.
    - A5 Data Generation (Random Sampling): $\{y_i, x_i\}$ are iid (or stationary and weakly dependent).
    - Distributional assumptions on ϵ_i hold (e.g., normal or logistic, independent of x_i).
- **Asymptotic Efficiency**:
  - Under these assumptions, Probit and Logit estimators are asymptotically efficient with variance:

$$I(\beta_0)^{-1} = \left[E\left(\frac{(f_\epsilon(x_i\beta_0))^2}{F_\epsilon(x_i\beta_0)(1 - F_\epsilon(x_i\beta_0))} x_i'x_i\right)\right]^{-1},$$

  where $f_\epsilon(x_i\beta_0)$ is the PDF (derivative of the CDF).

Interpretation of Binary Response Models

Binary response models, such as Probit and Logit, estimate the probability of an event occurring ($y_i = 1$) given predictor variables x_i. However, interpreting the estimated coefficients (β) in these models differs significantly from linear models. Below, we explore how to interpret these coefficients and the concept of **partial effects**.

1. Interpreting β in Binary Response Models

In binary response models, the coefficient β_j represents the average change in the **latent variable** y_i^* (an unobserved variable) for a one-unit change in x_{ij}. While this provides insight into the direction of the relationship:

- **Magnitudes** of β_j do not have a direct, meaningful interpretation in terms of y_i.
- **Direction** of β_j is meaningful:
  - $\beta_j > 0$: A positive association between x_{ij} and the probability of $y_i = 1$.
  - $\beta_j < 0$: A negative association between x_{ij} and the probability of $y_i = 1$.

2. Partial Effects in Nonlinear Binary Models

To interpret the effect of a change in a predictor x_{ij} on the probability of an event occurring ($P(y_i = 1|x_i)$), we use the **partial effect**:

$$E(y_i|x_i) = F_\epsilon(x_i\beta),$$

where $F_\epsilon(.)$ is the cumulative distribution function (CDF) of the error term ϵ_i (e.g., standard normal for Probit, logistic for Logit). The **partial effect** is the derivative of the expected probability with respect to x_{ij}:

$$PE(x_{ij}) = \frac{\partial E(y_i|x_i)}{\partial x_{ij}} = f_\epsilon(x_i\beta)\beta_j,$$

where:

- $f_\epsilon(.)$ is the probability density function (PDF) of the error term ϵ_i.
- β_j is the coefficient associated with x_{ij}.

3. Key Characteristics of Partial Effects

- **Scaling Factor**:
  - The partial effect depends on a **scaling factor**, $f_\epsilon(x_i\beta)$, which is derived from the density function $f_\epsilon(.)$.
  - The scaling factor varies depending on the values of x_i, making the partial effect nonlinear and **context-dependent**.
- **Non-Constant Partial Effects**:
  - Unlike linear models where coefficients directly represent constant marginal effects, the partial effect in binary models changes based on x_i.
  - For example, in a Logit model, the partial effect is largest when $P(y_i = 1|x_i)$ is around 0.5 (the midpoint of the S-shaped logistic curve) and smaller at the extremes (close to 0 or 1).

4. Single Values for Partial Effects

In practice, researchers often summarize partial effects using either:

- **Partial Effect at the Average (PEA)**:
  - The partial effect is calculated for an "average individual," where $x_i = \bar{x}$ (the sample mean of predictors):

$$PEA = f_\epsilon(\bar{x}\hat{\beta})\hat{\beta}_j.$$

  - This provides a single, interpretable value but assumes the average effect applies to all individuals.
- **Average Partial Effect (APE)**:
  - The average of all individual-level partial effects across the sample:

$$APE = \frac{1}{n}\sum_{i=1}^{n} f_\epsilon(x_i\hat{\beta})\hat{\beta}_j.$$

  - This accounts for the nonlinearity of the partial effects and provides a more accurate summary of the marginal effect in the population.

5. Comparing Partial Effects in Linear and Nonlinear Models

- **Linear Models**:
  - Partial effects are constant: $APE = PEA$.
  - The coefficients directly represent the marginal effects on $E(y_i|x_i)$.
- **Nonlinear Models**:
  - Partial effects are **not constant** due to the dependence on $f_\epsilon(x_i\beta)$.
  - As a result, $APE \neq PEA$ in general.

5.4 Penalized (Regularized) Estimators

Penalized or regularized estimators are extensions of Ordinary Least Squares designed to address its limitations, particularly in high-dimensional settings. Regularization methods introduce a penalty term to the loss function to prevent overfitting, handle multicollinearity, and improve model interpretability.

There are three popular regularization techniques (but not limited to):

1. Ridge Regression
2. Lasso Regression
3. Elastic Net

5.4.1 Motivation for Penalized Estimators

OLS minimizes the Residual Sum of Squares (RSS):

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - x_i'\beta)^2,$$

where:

- y_i is the observed outcome,
- x_i is the vector of predictors for observation i,
- β is the vector of coefficients.

While OLS works well under ideal conditions (e.g., low dimensionality, no multicollinearity), it struggles when:

- **Multicollinearity**: Predictors are highly correlated, leading to large variances in β estimates.
- **High Dimensionality**: The number of predictors (p) exceeds or approaches the sample size (n), making OLS inapplicable or unstable.
- **Overfitting**: When p is large, OLS fits noise in the data, reducing generalizability.

To address these issues, **penalized regression** modifies the OLS loss function by adding a **penalty term** that shrinks the coefficients toward zero. This discourages overfitting and improves predictive performance.

The general form of the penalized loss function is:

$$L(\beta) = \sum_{i=1}^{n} (y_i - x_i'\beta)^2 + \lambda P(\beta),$$

where:

- $\lambda \geq 0$: Tuning parameter controlling the strength of regularization.
- $P(\beta)$: Penalty term that quantifies model complexity.

Different choices of $P(\beta)$ lead to ridge regression, lasso regression, or elastic net.

5.4.2 Ridge Regression

Ridge regression, also known as **L2 regularization**, penalizes the sum of squared coefficients:

$$P(\beta) = \sum_{j=1}^{p} \beta_j^2.$$

The ridge objective function becomes:

$$L_{ridge}(\beta) = \sum_{i=1}^{n} (y_i - x_i'\beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2,$$

where:

- $\lambda \geq 0$ controls the degree of shrinkage. Larger λ leads to greater shrinkage.

Ridge regression has a closed-form solution:

$$\hat{\beta}_{ridge} = (X'X + \lambda I)^{-1} X'y,$$

where I is the $p \times p$ identity matrix.

Key Features

- Shrinks coefficients but does **not set them exactly to zero**.
- Handles multicollinearity effectively by stabilizing the coefficient estimates (Hoerl and Kennard, 1970).
- Works well when all predictors contribute to the response.

Example Use Case

Ridge regression is ideal for applications with many correlated predictors, such as:

- Predicting housing prices based on a large set of features (e.g., size, location, age of the house).

5.4.3 Lasso Regression

Lasso regression, or **L1 regularization**, penalizes the sum of absolute coefficients:

$$P(\beta) = \sum_{j=1}^{p} |\beta_j|.$$

The lasso objective function is:

$$L_{lasso}(\beta) = \sum_{i=1}^{n} (y_i - x_i'\beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j|.$$

Key Features

- Unlike ridge regression, lasso can set coefficients to **exactly zero**, performing automatic feature selection.

- Encourages sparse models, making it suitable for high-dimensional data (Tibshirani, 1996).

Optimization

Lasso does not have a closed-form solution due to the non-differentiability of $|\beta_j|$ at $\beta_j = 0$. It requires iterative algorithms, such as:

- **Coordinate Descent**,
- **Least Angle Regression (LARS)**.

Example Use Case

Lasso regression is useful when many predictors are irrelevant, such as:

- Genomics, where only a subset of genes are associated with a disease outcome.

5.4.4 Elastic Net

Elastic Net combines the penalties of ridge and lasso regression:

$$P(\beta) = \alpha \sum_{j=1}^{p} |\beta_j| + \frac{1-\alpha}{2} \sum_{j=1}^{p} \beta_j^2,$$

where:

- $0 \leq \alpha \leq 1$ determines the balance between lasso (L1) and ridge (L2) penalties.
- λ controls the overall strength of regularization.

The elastic net objective function is:

$$L_{elastic\ net}(\beta) = \sum_{i=1}^{n} (y_i - x_i'\beta)^2 + \lambda \left(\alpha \sum_{j=1}^{p} |\beta_j| + \frac{1-\alpha}{2} \sum_{j=1}^{p} \beta_j^2 \right).$$

Key Features

- Combines the strengths of lasso (sparse models) and ridge (stability with correlated predictors) (Zou and Hastie, 2005).
- Effective when predictors are highly correlated or when $p > n$.

Example Use Case

Elastic net is ideal for high-dimensional datasets with correlated predictors, such as:

- Predicting customer churn using demographic and behavioral features.

5.4.5 Tuning Parameter Selection

Choosing the regularization parameter λ (and α for elastic net) is critical for balancing model complexity (fit) and regularization (parsimony). If λ is too large, coefficients are overly shrunk (or even set to zero in the case of L1 penalty), leading to underfitting. If λ is too small, the model might overfit because coefficients are not penalized sufficiently. Hence, a systematic approach is needed to determine the optimal λ. For elastic net, we also choose an appropriate α to balance the L1 and L2 penalties.

5.4.5.1 Cross-Validation

A common approach to selecting λ (and α) is K-Fold Cross-Validation:

1. **Partition the data** into K roughly equal-sized "folds."
2. **Train the model** on $K-1$ folds and **validate** on the remaining fold, computing a validation error.
3. Repeat this process **for all folds**, and compute the average validation error across the K folds.
4. **Select** the value of λ (and α if tuning it) that **minimizes** the cross-validated error.

This method helps us maintain a good bias-variance trade-off because every point is used for both training and validation exactly once.

5.4.5.2 Information Criteria

Alternatively, one can use **information criteria**—like the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC)—to guide model selection. These criteria reward goodness-of-fit while penalizing model complexity, thereby helping in selecting an appropriately regularized model.

5.4.6 Properties of Penalized Estimators

1. **Bias-Variance Tradeoff**:
   - Regularization introduces some bias in exchange for reducing variance, often resulting in better predictive performance on new data.
2. **Shrinkage**:
   - Ridge shrinks coefficients toward zero but usually retains all predictors.
   - Lasso shrinks some coefficients exactly to zero, performing inherent feature selection.
3. **Flexibility**:

- Elastic net allows for a continuum between ridge and lasso, so it can adapt to different data structures (e.g., many correlated features or very high-dimensional feature spaces).

```r
# Load required libraries
library(glmnet)

# Simulate data
set.seed(123)
n <- 100   # Number of observations
p <- 20    # Number of predictors
X <- matrix(rnorm(n * p), nrow = n, ncol = p)  # Predictor matrix
y <- rnorm(n)                                  # Response vector

# Ridge regression (alpha = 0)
ridge_fit <- glmnet(X, y, alpha = 0)
plot(ridge_fit, xvar = "lambda", label = TRUE)
title("Coefficient Paths for Ridge Regression")
```

- In this plot, each curve represents a coefficient's value as a function of λ.
- As λ increases (moving from left to right on a log-scale by default), coefficients shrink toward zero but typically stay non-zero.
- Ridge regression tends to shrink coefficients but does not force them to be exactly zero.

```r
# Lasso regression (alpha = 1)
lasso_fit <- glmnet(X, y, alpha = 1)
plot(lasso_fit, xvar = "lambda", label = TRUE)
title("Coefficient Paths for Lasso Regression")
```

Here, as λ grows, several coefficient paths **hit zero exactly**, illustrating the variable selection property of lasso.

```r
# Elastic net (alpha = 0.5)
elastic_net_fit <- glmnet(X, y, alpha = 0.5)
plot(elastic_net_fit, xvar = "lambda", label = TRUE)
title("Coefficient Paths for Elastic Net (alpha = 0.5)")
```

- Elastic net combines ridge and lasso penalties. At $\lambda = 0.5$, we see partial shrinkage and some coefficients going to zero.

- This model is often helpful when you suspect both group-wise shrinkage (like ridge) and sparse solutions (like lasso) might be beneficial.

We can further refine our choice of λ by performing cross-validation on the lasso model:

```
cv_lasso <- cv.glmnet(X, y, alpha = 1)
plot(cv_lasso)
```

```
best_lambda <- cv_lasso$lambda.min
best_lambda
#> [1] 0.1449586
```

- The plot displays the cross-validated error (often mean-squared error or deviance) on the y-axis versus $\log(\lambda)$ on the x-axis.
- Two vertical dotted lines typically appear:
  1. $\lambda.min$: The λ that achieves the minimum cross-validated error.
  2. $\lambda.1se$: The largest λ such that the cross-validated error is still within one standard error of the minimum. This is a more conservative choice that favors higher regularization (simpler models).
- `best_lambda` above prints the numeric value of $\lambda.min$. This is the λ that gave the lowest cross-validation error for the lasso model.

Interpretation:

- By using `cv.glmnet`, we systematically compare different values of λ in terms of their predictive performance (cross-validation error).
- The selected λ typically balances having a smaller model (due to regularization) with retaining sufficient predictive power.
- If we used real-world data, we might also look at performance metrics on a hold-out test set to ensure that the chosen λ generalizes well.

5.5 Robust Estimators

Robust estimators are statistical techniques designed to provide reliable parameter estimates even when the assumptions underlying classical methods, such as Ordinary Least Squares, are violated. Specifically, they address issues caused by outliers, non-normal errors, or heavy-tailed distributions, which can render OLS inefficient or biased.

The goal of robust estimation is to reduce the sensitivity of the estimator to extreme or aberrant data points, thereby ensuring a more reliable and accurate fit to the majority of the data.

We will cover the key robust estimation techniques, their properties, and applications, along with practical examples and mathematical derivations. The focus will include M-estimators, R-estimators, L-estimators, LTS, S-estimators, MM-estimators, and more.

5.5.1 Motivation for Robust Estimation

OLS seeks to minimize the Residual Sum of Squares (RSS):

$$RSS = \sum_{i=1}^{n} (y_i - x_i'\beta)^2,$$

where:

- y_i is the observed response for the ith observation,
- x_i is the vector of predictors for the ith observation,
- β is the vector of coefficients.

OLS assumes:

1. Errors are normally distributed and no outliers in the data (A6 Normal Distribution).
2. Homoscedasticity (constant variance of errors) (A4 Homoskedasticity).

In real-world scenarios:

- **Outliers** in y or x can disproportionately affect the estimates, leading to biased or inefficient results.
- **Heavy-tailed distributions** (e.g., Cauchy) violate the normality assumption, making OLS inappropriate.

For example, Huber (1964) demonstrates that a single extreme observation can arbitrarily distort OLS estimates, while Hampel et al. (2005) define the breakdown point as a measure of robustness. Robust estimators aim to mitigate these problems by limiting the influence of problematic observations.

OLS inherently squares the residuals $e_i = y_i - x_i'\beta$, amplifying the influence of large residuals. For example, if a single residual is much larger than the others, its squared value can dominate the RSS, distorting the estimated coefficients.

Consider a simple case where $y_i = \beta_0 + \beta_1 x_i + e_i$, with $e_i \sim N(0, \sigma^2)$ under the classical assumptions. Now introduce an outlier: a single observation with an unusually large e_i. The squared residual for this point will dominate the RSS and pull the estimated regression line towards it, leading to biased estimates of β_0 and β_1.

The **breakdown point** of an estimator is the proportion of contamination (e.g., outliers) that the estimator can tolerate before yielding arbitrarily large or incorrect results. For OLS, the breakdown point is $1/n$, meaning even one outlier can cause substantial distortion in the estimates.

5.5.2 M-Estimators

To address the sensitivity of OLS, robust estimators minimize a different objective function:

$$\sum_{i=1}^{n} \rho\left(\frac{y_i - x_i'\beta}{\sigma}\right),$$

where:

- $\rho(\cdot)$ is a robust loss function that grows slower than the quadratic function used in OLS,
- σ is a scale parameter to normalize residuals.

In OLS, the quadratic loss function $\rho(z) = z^2$ penalizes large residuals disproportionately. Robust estimators replace this with alternative ρ functions that limit the penalty for large residuals, thus reducing their influence on the parameter estimates.

A robust ρ function should satisfy the following properties:

1. **Bounded Influence**: Large residuals contribute a finite amount to the objective function.
2. **Symmetry**: $\rho(z) = \rho(-z)$ ensures that positive and negative residuals are treated equally.
3. **Differentiability**: For computational tractability, ρ should be smooth and differentiable.

5.5.2.1 Examples of Robust ρ Functions

1. **Huber's Loss Function** (Huber, 1964)

Huber's loss function transitions between quadratic and linear growth:

$$\rho(z) = \begin{cases} \frac{z^2}{2} & \text{if } |z| \leq c, \\ c|z| - \frac{c^2}{2} & \text{if } |z| > c. \end{cases}$$

Key features:

- For small residuals ($|z| \leq c$), the loss is quadratic, mimicking OLS.
- For large residuals ($|z| > c$), the loss grows linearly, limiting their influence.

The parameter c controls the threshold at which the loss function transitions from quadratic to linear. Smaller values of c make the estimator more robust but potentially less efficient under normality.

2. **Tukey's Bisquare Function** (Beaton and Tukey, 1974)

Tukey's bisquare function completely bounds the influence of large residuals:

$$\rho(z) = \begin{cases} c^2 \left(1 - \left(1 - \left(\frac{z}{c}\right)^2\right)^3\right)/6 & \text{if } |z| \leq c, \\ c^2/6 & \text{if } |z| > c. \end{cases}$$

Key features:

- Residuals larger than c contribute a constant value to the objective function, effectively excluding them from the estimation process.
- This approach achieves high robustness at the cost of lower efficiency for small residuals.

3. **Andrews' Sine Function** (Andrews, 1974):
   - Smoothly downweights extreme residuals:

$$\rho(z) = \begin{cases} c^2 \left(1 - \cos\left(\frac{z}{c}\right)\right)/2 & \text{if } |z| \leq \pi c, \\ c^2/2 & \text{if } |z| > \pi c. \end{cases}$$

5.5.2.2 Weighting Scheme: Influence Functions

A critical concept in robust estimation is the **influence function**, which describes the sensitivity of the estimator to individual observations. For M-estimators, the influence function is derived as the derivative of the loss function $\rho(z)$ with respect to z:

$$\psi(z) = \frac{d}{dz}\rho(z).$$

This function plays a crucial role in downweighting large residuals. The weight assigned to each residual is proportional to $\psi(z)/z$, which decreases as $|z|$ increases for robust estimators.

For Huber's loss function, the influence function is:

$$\psi(z) = \begin{cases} z & \text{if } |z| \leq c, \\ c \cdot \text{sign}(z) & \text{if } |z| > c. \end{cases}$$

- For small residuals, $\psi(z) = z$, matching OLS.
- For large residuals, $\psi(z)$ is constant, ensuring bounded influence.

A key consideration when selecting a robust estimator is the trade-off between **robustness** (resistance to outliers) and **efficiency** (performance under ideal conditions). The tuning parameters in ρ functions (e.g., c in Huber's loss) directly affect this balance:

- Smaller c increases robustness but reduces efficiency under normality.
- Larger c improves efficiency under normality but decreases robustness to outliers.

This trade-off reflects the fundamental goal of robust estimation: to achieve a balance between reliability and precision across a wide range of data scenarios.

5.5.2.3 Properties of M-Estimators

Robust estimators, particularly M-estimators, possess the following mathematical properties:

1. **Asymptotic Normality**: Under mild regularity conditions, M-estimators are asymptotically normal:

$$\sqrt{n}(\hat{\beta} - \beta) \xrightarrow{d} N(0, \Sigma),$$

   where Σ depends on the choice of ρ and the distribution of residuals.
2. **Consistency**: As $n \to \infty$, $\hat{\beta} \to \beta$ in probability, provided the majority of the data satisfies the model assumptions.
3. **Breakdown Point**: M-estimators typically have a moderate breakdown point, sufficient to handle a reasonable proportion of contamination.

5.5.3 R-Estimators

R-estimators are a class of robust estimators that rely on the ranks of residuals rather than their raw magnitudes. This approach makes them naturally resistant to the influence of outliers and highly effective in scenarios involving ordinal

data or heavy-tailed error distributions. By leveraging rank-based methods, R-estimators are particularly useful in situations where classical assumptions about the data, such as normality or homoscedasticity, do not hold.

The general form of an R-estimator can be expressed as:

$$\hat{\beta}_R = \arg\min_{\beta} \sum_{i=1}^{n} w_i R_i \left(y_i - x_i'\beta\right),$$

where:

- R_i are the ranks of residuals $e_i = y_i - x_i'\beta$,
- w_i are rank-based weights determined by a chosen scoring function,
- y_i are observed responses, x_i are predictor values, and β is the vector of coefficients.

This formulation differs from M-estimators, which directly minimize a loss function ρ, by instead using the **ordering of residuals** to drive the estimation.

5.5.3.1 Ranks and Scoring Function

5.5.3.1.1 Definition of Ranks The rank R_i of a residual e_i is its position in the sorted sequence of all residuals:

$$R_i = \text{rank}(e_i) = \sum_{j=1}^{n} \mathbb{I}(e_j \leq e_i),$$

where $\mathbb{I}(\cdot)$ is the indicator function, equal to 1 if the condition is true and 0 otherwise. This step transforms the residuals into an ordinal scale, eliminating their dependency on magnitude.

5.5.3.1.2 Scoring Function The weights w_i are derived from a scoring function $S(R_i)$, which assigns importance to each rank. A common choice is the **Wilcoxon scoring function**, defined as:

$$S(R_i) = \frac{R_i}{n+1},$$

which gives equal weight to all ranks, scaled by their position relative to the total number of observations n.

Other scoring functions can emphasize different parts of the rank distribution:

- **Normal Scores**: Derived from the quantiles of a standard normal distribution.
- **Logarithmic Scores**: Weight lower ranks more heavily.

The flexibility of the scoring function allows R-estimators to adapt to various data structures and assumptions.

5.5.3.2 Properties of R-Estimators

5.5.3.2.1 Influence Function and Robustness A key feature of R-estimators is their **bounded influence function**, which ensures robustness. Because the estimator depends only on the ranks of the residuals, extreme values in y or x do not disproportionately affect the results.

For R-estimators, the influence function $\psi(e_i)$ is proportional to the derivative of the rank-based objective function:

$$\psi(e_i) = S'(R_i),$$

where $S'(R_i)$ is the derivative of the scoring function. Since R_i depends only on the ordering of residuals, outliers in the data cannot produce excessive changes in R_i, resulting in bounded influence.

5.5.3.2.2 Breakdown Point The **breakdown point** of R-estimators is higher than that of OLS and comparable to other robust methods. This means they can tolerate a larger proportion of contaminated data without yielding unreliable results.

5.5.3.2.3 Asymptotic Efficiency Under specific scoring functions, R-estimators achieve high asymptotic efficiency. For example, the Wilcoxon R-estimator performs nearly as well as OLS under normality while retaining robustness to non-normality.

5.5.3.3 Derivation of R-Estimators for Simple Linear Regression

Consider the simple linear regression model:

$$y_i = \beta_0 + \beta_1 x_i + e_i,$$

where $e_i = y_i - (\beta_0 + \beta_1 x_i)$ are the residuals.

1. **Rank the Residuals**: Compute the residuals e_i for all observations and rank them from smallest to largest.
2. **Assign Weights**: Compute weights w_i for each residual rank based on the scoring function $S(R_i)$.

3. **Minimize the Rank-Based Objective**: Solve the following optimization problem:

$$\hat{\beta}_R = \arg\min_{\beta_0,\beta_1} \sum_{i=1}^{n} w_i R_i \left(y_i - (\beta_0 + \beta_1 x_i)\right).$$

This minimization can be performed iteratively using numerical methods, as the rank-based nature of the function makes direct analytic solutions challenging.

5.5.3.4 Comparison to M-Estimators

While M-estimators downweight large residuals using robust loss functions, R-estimators completely avoid reliance on the magnitude of residuals by using their ranks. This distinction has important implications:

- R-estimators are naturally robust to leverage points and extreme outliers.
- The performance of R-estimators is less sensitive to the choice of scale parameter compared to M-estimators.
- However, R-estimators may be less efficient than M-estimators under normality because they do not use the full information contained in the residual magnitudes.

5.5.4 L-Estimators

L-estimators are a class of robust estimators constructed as **linear combinations of order statistics**, where order statistics are simply the sorted values of a dataset. These estimators are particularly appealing due to their intuitive nature and computational simplicity. By using the relative ranks of observations, L-estimators offer robustness against outliers and heavy-tailed distributions.

Order statistics are denoted as $y_{(1)}, y_{(2)}, \dots, y_{(n)}$, where $y_{(i)}$ is the ith smallest observation in the sample.

The general form of an L-estimator is:

$$\hat{\theta}_L = \sum_{i=1}^{n} c_i y_{(i)},$$

where:

- $y_{(i)}$ are the order statistics (sorted observations),

- c_i are coefficients (weights) that determine the contribution of each order statistic to the estimator.

By appropriately choosing the weights c_i, different types of L-estimators can be constructed to suit specific needs, such as handling outliers or capturing central tendencies robustly.

Examples of L-Estimators

1. **Sample Median**: The sample median is a simple L-estimator where only the middle order statistic contributes (for odd n) or the average of the two middle order statistics contributes (for even n):

$$\hat{\mu}_{\text{median}} = \begin{cases} y_{(\frac{n+1}{2})} & \text{if } n \text{ is odd,} \\ \frac{1}{2}\left(y_{(\frac{n}{2})} + y_{(\frac{n}{2}+1)}\right) & \text{if } n \text{ is even.} \end{cases}$$

   - **Robustness**: The median has a breakdown point of 50%, meaning it remains unaffected unless more than half the data are corrupted.
   - **Efficiency**: Under normality, the efficiency of the median is lower than that of the mean (about 64%).

2. **Trimmed Mean**: The trimmed mean excludes the smallest and largest k% of observations before averaging the remaining values:

$$\hat{\mu}_T = \frac{1}{n-2k} \sum_{i=k+1}^{n-k} y_{(i)},$$

   where:

   - k is the number of observations trimmed from each tail,
   - n is the sample size.
   - **Robustness**: The trimmed mean is less sensitive to extreme values than the sample mean.
   - **Efficiency**: By retaining most observations, the trimmed mean achieves a good balance between robustness and efficiency.

3. **Winsorized Mean**: Similar to the trimmed mean, but instead of excluding extreme values, it replaces them with the nearest remaining observations:

$$\hat{\mu}_W = \frac{1}{n} \sum_{i=1}^{n} y^*_{(i)},$$

where $y^*_{(i)}$ are "Winsorized" values:

$$y^*_{(i)} = \begin{cases} y_{(k+1)} & \text{if } i \leq k, \\ y_{(i)} & \text{if } k+1 \leq i \leq n-k, \\ y_{(n-k)} & \text{if } i > n-k. \end{cases}$$

- **Robustness**: The Winsorized mean reduces the influence of outliers without discarding data.
- **Efficiency**: Slightly less efficient than the trimmed mean under normality.

4. **Midrange**: The midrange is the average of the smallest and largest observations:

$$\hat{\mu}_{\text{midrange}} = \frac{y_{(1)} + y_{(n)}}{2}.$$

- **Robustness**: Poor robustness, as it depends entirely on the extreme observations.
- **Simplicity**: Highly intuitive and computationally trivial.

5.5.4.1 Properties of L-Estimators

1. **Robustness to Outliers**: L-estimators gain robustness by downweighting or excluding extreme observations. For instance:
   - The trimmed mean completely removes outliers from the estimation process.
   - The Winsorized mean limits the influence of outliers by bounding their values.
2. **Breakdown Point**:
   - The breakdown point of an L-estimator depends on how many extreme observations are excluded or replaced.
   - The median has the highest possible breakdown point (50%), while the trimmed and Winsorized means have breakdown points proportional to the trimming percentage.
3. **Efficiency**:
   - The efficiency of L-estimators varies depending on the underlying data distribution and the specific estimator.
   - For symmetric distributions, the trimmed mean and Winsorized mean approach the efficiency of the sample mean while being much more robust.
4. **Computational Simplicity**:
   - L-estimators involve simple operations like sorting and averaging, making them computationally efficient even for large datasets.

5.5.4.2 Derivation of the Trimmed Mean

To understand the robustness of the trimmed mean, consider a dataset with n observations. Sorting the data gives $y_{(1)} \leq y_{(2)} \leq \cdots \leq y_{(n)}$. After trimming the smallest k and largest k observations, the remaining $n-2k$ observations are used to compute the mean:

$$\hat{\mu}_T = \frac{1}{n-2k} \sum_{i=k+1}^{n-k} y_{(i)}.$$

Key observations:

- **Impact of** k: Larger k increases robustness by removing more extreme values but reduces efficiency by discarding more data.
- **Choosing** k: In practice, k is often chosen as a percentage of the total sample size, such as 10% trimming ($k = 0.1n$).

5.5.5 Least Trimmed Squares (LTS)

Least Trimmed Squares (LTS) is a robust regression method that minimizes the sum of the smallest h squared residuals, rather than using all residuals as in Ordinary Least Squares. This approach ensures that large residuals, often caused by outliers or leverage points, have no influence on the parameter estimation.

The LTS estimator is defined as:

$$\hat{\beta}_{LTS} = \arg\min_{\beta} \sum_{i=1}^{h} r_{[i]}^2,$$

where:

- $r_{[i]}^2$ are the ordered squared residuals, ranked from smallest to largest,
- h is the subset size of residuals to include in the minimization, typically chosen as $h = \lfloor n/2 \rfloor + 1$ (where n is the sample size).

This trimming process ensures robustness by focusing on the best-fitting h observations and ignoring the most extreme residuals.

5.5.5.1 Motivation for LTS

In OLS regression, the objective is to minimize the Residual Sum of Squares (RSS):

$$RSS = \sum_{i=1}^{n} r_i^2,$$

where $r_i = y_i - x_i'\beta$ are the residuals. However, this method is highly sensitive to outliers because even one large residual (r_i^2) can dominate the RSS, distorting the parameter estimates β.

LTS addresses this issue by trimming the largest residuals and focusing only on the h smallest ones, thus preventing extreme values from affecting the fit. This approach provides a more robust estimate of the regression coefficients β.

5.5.5.2 Properties of LTS

1. **Objective Function**: The LTS objective function is non-differentiable because it involves ordering the squared residuals. Formally, the ordered residuals are denoted as:

   $$r_{[1]}^2 \leq r_{[2]}^2 \leq \cdots \leq r_{[n]}^2,$$

   and the objective is to minimize:

   $$\sum_{i=1}^{h} r_{[i]}^2.$$

   This requires sorting the squared residuals, making the computation more complex than OLS.

2. **Choice of h**: The parameter h determines the number of residuals included in the minimization. A common choice is:

   $$h = \lfloor n/2 \rfloor + 1,$$

   which ensures a high breakdown point (discussed below). Smaller values of h increase robustness but reduce efficiency, while larger h values improve efficiency but decrease robustness.

3. **Breakdown Point**: LTS has a **breakdown point** of approximately 50%, the highest possible for a regression estimator. This means that LTS can handle up to 50% of contaminated data (e.g., outliers) without yielding unreliable estimates.

4. **Robustness**: By focusing only on the h best-fitting observations, LTS naturally excludes outliers from the estimation process, making it highly

robust to both vertical outliers (extreme values in y) and leverage points (extreme values in x).

5.5.5.3 Algorithm for LTS

Computing the LTS estimator involves the following steps:

1. **Initialization**: Select an initial subset of h observations to compute a preliminary fit for β.
2. **Residual Calculation**: For each observation, compute the squared residuals:

$$r_i^2 = (y_i - x_i'\beta)^2 .$$

3. **Trimming**: Rank the residuals from smallest to largest and retain only the h smallest residuals.
4. **Refitting**: Use the h retained observations to recompute the regression coefficients β.
5. **Iterative Refinement**: Repeat the process (residual calculation, trimming, refitting) until convergence, typically when β stabilizes.

Efficient algorithms, such as the **Fast-LTS** algorithm, are used in practice to reduce computational complexity.

5.5.5.4 Comparison of LTS with OLS

Property	OLS	LTS
Objective	Minimize $\sum_{i=1}^{n} r_i^2$	Minimize $\sum_{i=1}^{h} r_{[i]}^2$
Sensitivity to Outliers	High	Low
Breakdown Point	$1/n$	$\approx 50\%$
Computational Cost	Low	Moderate (requires sorting and iterations)

5.5.6 S-Estimators

S-estimators are a class of robust estimators that focus on minimizing a robust measure of the **dispersion** of residuals. Unlike methods such as M-estimators, which directly minimize a loss function based on residuals, S-estimators aim

to find the parameter values β that produce residuals with the smallest robust scale. These estimators are particularly useful in handling datasets with outliers, heavy-tailed distributions, or other violations of classical assumptions.

The scale σ is estimated by solving the following minimization problem:

$$\hat{\sigma}_S = \arg\min_{\sigma} \frac{1}{n} \sum_{i=1}^{n} \rho \left(\frac{y_i - x_i'\beta}{\sigma} \right),$$

where:

- ρ is a robust loss function that controls the influence of residuals,
- y_i are observed responses, x_i are predictors, β is the vector of regression coefficients,
- σ represents the robust scale of the residuals.

Once σ is estimated, the S-estimator of β is obtained by solving:

$$\hat{\beta}_S = \arg\min_{\beta} \hat{\sigma}_S.$$

5.5.6.1 Motivation for S-Estimators

In regression analysis, classical methods such as Ordinary Least Squares rely on minimizing the Residual Sum of Squares (RSS). However, OLS is highly sensitive to outliers because even a single extreme residual can dominate the sum of squared residuals, leading to biased estimates of β.

S-estimators address this limitation by using a robust scale σ to evaluate the dispersion of residuals. By minimizing this scale, S-estimators effectively down-weight the influence of outliers, resulting in parameter estimates that are more resistant to contamination in the data.

5.5.6.2 Key Concepts in S-Estimators

1. **Robust Scale Function**: The key idea of S-estimators is to minimize a robust measure of scale. The scale σ is computed such that the residuals normalized by σ produce a value close to the expected contribution of well-behaved observations.

   Formally, σ satisfies:

$$\frac{1}{n} \sum_{i=1}^{n} \rho \left(\frac{y_i - x_i'\beta}{\sigma} \right) = \delta,$$

where δ is a constant that depends on the choice of ρ and ensures consistency under normality. This equation balances the residuals and controls their influence on the scale estimate.

2. **Choice of** ρ-Function: The choice of the robust ρ function is critical in determining the behavior of S-estimators. Common ρ functions include:

   - **Huber's** ρ-Function:

$$\rho(z) = \begin{cases} z^2/2 & \text{if } |z| \leq c, \\ c|z| - c^2/2 & \text{if } |z| > c. \end{cases}$$

   - **Tukey's Bisquare**:

$$\rho(z) = \begin{cases} c^2 \left(1 - \left(1 - \left(\frac{z}{c}\right)^2\right)^3\right)/6 & \text{if } |z| \leq c, \\ c^2/6 & \text{if } |z| > c. \end{cases}$$

   - **Andrews' Sine**:

$$\rho(z) = \begin{cases} c^2 \left(1 - \cos\left(\frac{z}{c}\right)\right)/2 & \text{if } |z| \leq \pi c, \\ c^2/2 & \text{if } |z| > \pi c. \end{cases}$$

   Robust ρ functions grow more slowly than the quadratic function used in OLS, limiting the impact of large residuals.

5.5.6.3 Properties of S-Estimators

1. **Breakdown Point**: S-estimators have a breakdown point of up to 50%, meaning they can tolerate up to half the data being contaminated (e.g., outliers) without yielding unreliable estimates.

2. **Efficiency**: The efficiency of S-estimators depends on the choice of ρ. While they are highly robust, their efficiency under ideal conditions (e.g., normality) may be lower than that of OLS. Proper tuning of ρ can balance robustness and efficiency.

3. **Influence Function**: The **influence function** measures the sensitivity of the estimator to a small perturbation in the data. For S-estimators, the influence function is bounded, ensuring robustness to outliers.

4. **Consistency**: Under mild regularity conditions, S-estimators are consistent, meaning $\widehat{\beta}_S \to \beta$ as the sample size $n \to \infty$.

5. **Asymptotic Normality**: S-estimators are asymptotically normal, with:

$$\sqrt{n}(\widehat{\beta}_S - \beta) \xrightarrow{d} N(0, \Sigma),$$

where Σ depends on the choice of ρ and the distribution of residuals.

5.5.6.4 Algorithm for Computing S-Estimators

1. **Initial Guess**: Compute an initial estimate of β using a robust method (e.g., LTS or an M-estimator).
2. **Scale Estimation**: Compute a robust estimate of scale $\hat{\sigma}$ by solving:

$$\frac{1}{n}\sum_{i=1}^{n}\rho\left(\frac{y_i - x_i'\beta}{\sigma}\right) = \delta.$$

3. **Iterative Refinement**:
   - Recalculate residuals $r_i = y_i - x_i'\beta$.
   - Update β and σ iteratively until convergence, typically using numerical optimization techniques.

5.5.7 MM-Estimators

MM-estimators are a robust regression method that combines the strengths of two powerful techniques: S-estimators and M-estimators. They are designed to achieve both a **high breakdown point** (up to 50%) and **high efficiency** under ideal conditions (e.g., normality). This combination makes MM-estimators one of the most versatile and widely used robust regression methods.

The process of computing MM-estimators involves three main steps:

1. Compute an initial robust estimate of scale using an S-estimator.
2. Use this robust scale to define weights for an M-estimator.
3. Estimate regression coefficients by solving the weighted M-estimation problem.

This stepwise approach ensures robustness in the initial scale estimation while leveraging the efficiency of M-estimators for the final parameter estimates.

Step 1: Robust Scale Estimation

The first step is to estimate the robust scale σ using an S-estimator. This involves solving:

$$\hat{\sigma}_S = \arg\min_{\sigma} \frac{1}{n}\sum_{i=1}^{n}\rho_S\left(\frac{y_i - x_i'\beta}{\sigma}\right),$$

where ρ_S is a robust loss function chosen to control the influence of extreme residuals. Common choices for ρ_S include Huber's or Tukey's bisquare functions. This scale estimation provides a robust baseline for weighting residuals in the subsequent M-estimation step.

Step 2: Weight Definition for M-Estimation

Using the robust scale $\hat{\sigma}_S$ obtained in Step 1, the weights for the M-estimator are defined based on a second loss function, ρ_M. The weights downweight residuals proportional to their deviation relative to $\hat{\sigma}_S$. For each residual $r_i = y_i - x_i'\beta$, the weight is computed as:

$$w_i = \psi_M\left(\frac{r_i}{\hat{\sigma}_S}\right) / \frac{r_i}{\hat{\sigma}_S},$$

where:

- ψ_M is the derivative of the robust ρ_M function, known as the **influence function**.
- ρ_M is often chosen to provide high efficiency under normality, such as Huber's or Hampel's function.

These weights reduce the impact of large residuals while preserving the influence of small, well-behaved residuals.

Step 3: Final M-Estimation

The final step involves solving the M-estimation problem using the weights defined in Step 2. The coefficients $\hat{\beta}_{MM}$ are estimated by minimizing the weighted residuals:

$$\hat{\beta}_{MM} = \arg\min_{\beta} \sum_{i=1}^{n} w_i \rho_M\left(\frac{y_i - x_i'\beta}{\hat{\sigma}_S}\right).$$

This ensures that the final estimates combine the robustness of the initial S-estimator with the efficiency of the M-estimator.

5.5.7.1 Properties of MM-Estimators

1. **High Breakdown Point**:
   - The S-estimator in the first step ensures a breakdown point of up to 50%, meaning the estimator can handle up to half the data being contaminated without producing unreliable results.
2. **Asymptotic Efficiency**:

- The use of an efficient ρ_M function in the final M-estimation step ensures that MM-estimators achieve high asymptotic efficiency under normality, often close to that of OLS.

3. **Robustness**:
   - The combination of robust scale estimation and downweighting of large residuals makes MM-estimators highly robust to outliers and leverage points.
4. **Influence Function**:
   - The influence function of MM-estimators is bounded, ensuring that no single observation can exert disproportionate influence on the parameter estimates.
5. **Consistency**:
   - MM-estimators are consistent, converging to the true parameter values as the sample size increases, provided the majority of the data satisfies the model assumptions.
6. **Asymptotic Normality**:
   - MM-estimators are asymptotically normal, with:

$$\sqrt{n}(\hat{\beta}_{MM} - \beta) \xrightarrow{d} N(0, \Sigma),$$

where Σ depends on the choice of ρ_M and the distribution of residuals.

5.5.7.2 Choice of ρ-Functions for MM-Estimators

The robustness and efficiency of MM-estimators depend on the choice of ρ_S (for scale) and ρ_M (for final estimation). Common choices include:

1. **Huber's** ρ-Function: Combines quadratic and linear growth to balance robustness and efficiency:

$$\rho(z) = \begin{cases} \frac{z^2}{2} & \text{if } |z| \leq c, \\ c|z| - \frac{c^2}{2} & \text{if } |z| > c. \end{cases}$$

2. **Tukey's Bisquare Function**: Provides high robustness by completely bounding large residuals:

$$\rho(z) = \begin{cases} c^2 \left(1 - \left(1 - \left(\frac{z}{c}\right)^2\right)^3\right)/6 & \text{if } |z| \leq c, \\ c^2/6 & \text{if } |z| > c. \end{cases}$$

3. **Hampel's Three-Part Redescending Function**: Further limits the influence of large residuals by assigning a constant penalty beyond a certain threshold.

$$\rho(z) = \begin{cases} z^2/2 & \text{if } |z| \leq a, \\ a|z| - a^2/2 & \text{if } a < |z| \leq b, \\ \text{constant} & \text{if } |z| > b. \end{cases}$$

5.5.8 Practical Considerations

The following table summarizes the key properties, advantages, and limitations of the robust estimators discussed: +————————+————————————————————————————————————-+————————————————-+——+——+ | Estimator | Key Features | Breakdown Point | Efficiency (Under Normality) | Applications | Advantages | +==========================+==

| M-Estimators | Generalization of OLS Robust ρ reduces large residual influence | Moderate (up to 0.29) | High with proper tuning | Wide applicability in regression with moderate robustness | Balances robustness and efficiency | | | | | | | | | | | | | | Flexible tuning via ρ-function | +————————+————————————————————————————————————-+————————————————-+——+——+ | R-Estimators | Rank-based method | High (depends on ranks) | Moderate | Ordinal data or heavily skewed distributions | Handles both predictor and response outliers | | | | | | | | | Immune to outliers in x and y | | | | Suitable for ordinal or rank-based data | +————————+————————————————————————————————————-+————————————————-+——+——+ | L-Estimators | Linear combination of order statistics | High (up to 50%) | Moderate | Descriptive statistics, robust averages | Simple and intuitive | | | | | | | | | | | | | | Easy to compute, even for large datasets | +————————+————————————————————————————————————-+————————————————-+——+——+ | **LTS** | Minimizes smallest h squared residuals | High (up to 50%) | Moderate | Data with high contamination, fault detection | High robustness to outliers | | | | | | | | | | | | | | Resistant to leverage points | +————————+————————————————————————————————————-+————————————————-+——+——+ | S-Estimators | Minimizes robust scale of residuals | High (up to 50%) | Low to moderate | Outlier detection, data with heavy-tailed distributions | Focus on robust scale estimation | | | | | | | | | | | | | | Effective at detecting extreme outliers | +————————+————————————————————————————————————-+————————————————-+——+——+ | MM-Estimators | High robustness (scale) + high efficiency (coefficients) | High (up to 50%) | High

| Real-world applications with mixed contamination and heavy-tailed errors | Combinesrobustness and efficiency effectively | | | | | | | | | | | | | Versatile and flexible | +————————+————————————————————————————-+————————————-+————————————————————————————————————+————————————————————————————————————+

Notes on Choosing an Estimator

- M-Estimators: Best suited for general-purpose robust regression, offering a balance between robustness and efficiency with moderate contamination.
- R-Estimators: Ideal for rank-based data or ordinal data, especially when outliers are present in both predictors and responses.
- L-Estimators: Simple and effective for descriptive statistics or data cleaning with limited computational resources.
- **LTS**: Recommended for datasets with significant contamination or leverage points due to its high breakdown point.
- S-Estimators: Focus on robust scale estimation, suitable for identifying and mitigating the influence of extreme residuals.
- MM-Estimators: Combines the robustness of S-estimators with the efficiency of M-estimators, making it the most versatile choice for heavily contaminated data.

```r
# Load necessary libraries
library(MASS)       # For robust regression functions like rlm
library(robustbase) # For LTS regression and MM-estimators
library(dplyr)      # For data manipulation
library(ggplot2)    # For visualization

# Simulate dataset
set.seed(123)
n <- 100
x <- rnorm(n, mean = 5, sd = 2)   # Predictor
y <- 3 + 2 * x + rnorm(n, sd = 1) # Response

# Introduce outliers
y[95:100] <- y[95:100] + 20  # Vertical outliers
x[90:95] <- x[90:95] + 10    # Leverage points

data <- data.frame(x, y)

# Visualize the data
ggplot(data, aes(x, y)) +
    geom_point() +
    labs(title = "Scatterplot of Simulated Data with Outliers",
```

```
        x = "Predictor (x)",
        y = "Response (y)") +
    theme_minimal()
```

```
# Ordinary Least Squares
ols_model <- lm(y ~ x, data = data)
summary(ols_model)
#>
#> Call:
#> lm(formula = y ~ x, data = data)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -12.6023  -2.4590  -0.5717   0.9247  24.4024
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   8.8346     1.1550   7.649 1.41e-11 ***
#> x             0.9721     0.1749   5.558 2.36e-07 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 5.583 on 98 degrees of freedom
#> Multiple R-squared:  0.2396, Adjusted R-squared:  0.2319
#> F-statistic: 30.89 on 1 and 98 DF,  p-value: 2.358e-07
```

OLS coefficients are highly influenced by the presence of outliers. For example,

the slope (x coefficient) and intercept are shifted to fit the outliers, resulting in a poor fit to the majority of the data.

```
# $M$-Estimators
m_model <- rlm(y ~ x, data = data, psi = psi.huber)
summary(m_model)
#>
#> Call: rlm(formula = y ~ x, data = data, psi = psi.huber)
#> Residuals:
#>       Min        1Q    Median        3Q       Max
#> -18.43919  -0.97575  -0.03297   0.76967  21.85546
#>
#> Coefficients:
#>             Value   Std. Error t value
#> (Intercept)  4.3229  0.2764    15.6421
#> x            1.7250  0.0419    41.2186
#>
#> Residual standard error: 1.349 on 98 degrees of freedom
```

The M-estimator reduces the influence of large residuals using Huber's psi function. This results in coefficients that are less affected by outliers compared to OLS.

```
# Least Trimmed Squares (LTS)
lts_model <- ltsReg(y ~ x, data = data)
lts_coefficients <- coef(lts_model)
```

LTS minimizes the smallest squared residuals, ignoring extreme residuals. This results in a more robust fit, particularly in the presence of both vertical outliers and leverage points.

```
# $MM$-Estimators
mm_model <- lmrob(y ~ x, data = data, setting = "KS2014")
summary(mm_model)
#>
#> Call:
#> lmrob(formula = y ~ x, data = data, setting = "KS2014")
#>  \--> method = "SMDM"
#> Residuals:
#>       Min        1Q    Median        3Q       Max
#> -20.45989  -0.69436  -0.01455   0.73614  22.10173
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  3.02192    0.25850   11.69   <2e-16 ***
#> x            1.96672    0.04538   43.34   <2e-16 ***
#> ---
```

```
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Robust residual standard error: 0.9458
#> Multiple R-squared:  0.9562, Adjusted R-squared:  0.9558
#> Convergence in 7 IRWLS iterations
#>
#> Robustness weights:
#>  10 observations c(90,91,92,93,94,96,97,98,99,100)
#>   are outliers with |weight| = 0 ( < 0.001);
#>  67 weights are ~= 1. The remaining 23 ones are summarized as
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.2496  0.7969  0.9216  0.8428  0.9548  0.9943
#> Algorithmic parameters:
#>       tuning.chi1       tuning.chi2       tuning.chi3       tuning.chi4
#>        -5.000e-01         1.500e+00                NA         5.000e-01
#>                bb       tuning.psi1       tuning.psi2       tuning.psi3
#>         5.000e-01        -5.000e-01         1.500e+00         9.500e-01
#>       tuning.psi4        refine.tol           rel.tol         scale.tol
#>                NA         1.000e-07         1.000e-07         1.000e-10
#>         solve.tol          zero.tol       eps.outlier             eps.x
#>         1.000e-07         1.000e-10         1.000e-03         3.223e-11
#> warn.limit.reject warn.limit.meanrw
#>         5.000e-01         5.000e-01
#>      nResample         max.it       best.r.s       k.fast.s          k.max
#>           1000            500             20              2           2000
#>    maxit.scale      trace.lev            mts     compute.rd      numpoints
#>            200              0           1000              0             10
#> fast.s.large.n
#>           2000
#>               setting                   psi           subsampling
#>              "KS2014"                 "lqq"         "nonsingular"
#>                   cov compute.outlier.stats
#>             ".vcov.w"                "SMDM"
#> seed : int(0)
```

MM-estimators combine robust scale estimation (from S-estimators) with efficient coefficient estimation (from M-estimators). This achieves both high robustness and high efficiency under normal conditions.

```
# Visualizing results
data <- data %>%
    mutate(
        ols_fit = predict(ols_model, newdata = data),
        m_fit = predict(m_model, newdata = data),
        lts_fit = fitted(lts_model),
        # Use `fitted()` for ltsReg objects
```

```
        mm_fit = predict(mm_model, newdata = data)
    )

ggplot(data, aes(x, y)) +
    geom_point() +
    geom_line(
        aes(y = ols_fit),
        color = "red",
        linetype = "dashed",
        size = 1,
        label = "OLS"
    ) +
    geom_line(
        aes(y = m_fit),
        color = "blue",
        linetype = "dashed",
        size = 1,
        label = "$M$-Estimator"
    ) +
    geom_line(
        aes(y = lts_fit),
        color = "green",
        linetype = "dashed",
        size = 1,
        label = "LTS"
    ) +
    geom_line(
        aes(y = mm_fit),
        color = "purple",
        linetype = "dashed",
        size = 1,
        label = "$MM$-Estimator"
    ) +
    labs(title = "Comparison of Regression Fits",
         x = "Predictor (x)",
         y = "Response (y)") +
    theme_minimal()
```

Visualization shows the differences in regression fits: - OLS is heavily influenced by outliers and provides a poor fit to the majority of the data.) - The M-estimator downweights large residuals, resulting in a better fit. - LTS regression ignores the extreme residuals entirely, providing the most robust fit. - MM-estimators balance robustness and efficiency, producing coefficients close to the LTS but with improved efficiency under normality.

```
# Comparing Coefficients
comparison <- data.frame(
    Method = c("OLS", "$M$-Estimator", "LTS", "$MM$-Estimator"),
    Intercept = c(
        coef(ols_model)[1],
        coef(m_model)[1],
        lts_coefficients[1],
        coef(mm_model)[1]
    ),
    Slope = c(
        coef(ols_model)[2],
        coef(m_model)[2],
        lts_coefficients[2],
        coef(mm_model)[2]
    )
)

print(comparison)
#>            Method Intercept     Slope
#> 1             OLS  8.834553 0.9720994
```

```
#> 2  $M$-Estimator  4.322869 1.7250441
#> 3            LTS  2.954960 1.9777635
#> 4 $MM$-Estimator  3.021923 1.9667208
```

The table above shows how the coefficients vary across methods: - OLS coefficients are the most distorted by outliers. - M-estimators and MM-estimators provide coefficients that are less influenced by extreme values. - LTS regression, with its trimming mechanism, produces the most robust coefficients by excluding the largest residuals.

5.6 Partial Least Squares

Partial Least Squares (PLS) is a dimensionality reduction technique used for regression and predictive modeling. It is particularly useful when predictors are highly collinear or when the number of predictors (p) exceeds the number of observations (n). Unlike methods such as Principal Component Regression (PCR), PLS simultaneously considers the relationship between predictors and the response variable.

5.6.1 Motivation for PLS

Limitations of Classical Methods

1. **Multicollinearity**:
   - OLS fails when predictors are highly correlated because the design matrix $X'X$ becomes nearly singular, leading to unstable estimates.
2. **High-Dimensional Data**:
   - When $p > n$, OLS cannot be directly applied as $X'X$ is not invertible.
3. **Principal Component Regression (PCR)**:
   - While PCR addresses multicollinearity by using principal components of X, it does not account for the relationship between predictors and the response variable y when constructing components.

PLS overcomes these limitations by constructing components that maximize the covariance between predictors X and the response y. It finds a compromise between explaining the variance in X and predicting y, making it particularly suited for regression in high-dimensional or collinear datasets.

Let:

- X be the $n \times p$ matrix of predictors,
- y be the $n \times 1$ response vector,
- t_k be the k-th latent component derived from X,

- p_k and q_k be the loadings for X and y, respectively.

PLS aims to construct latent components $t_1, t_2, \dots, t_K$ such that:

1. Each t_k is a linear combination of the predictors: $t_k = Xw_k$, where w_k is a weight vector. 2
2. The covariance between t_k and y is maximized:

$$\text{Maximize } Cov(t_k, y) = w_k' X' y.$$

5.6.2 Steps to Construct PLS Components

1. **Compute Weights**:
   - The weights w_k for the k-th component are obtained by solving:

$$w_k = \frac{X'y}{\|X'y\|}.$$

2. **Construct Latent Component**:
   - Form the k-th latent component:

$$t_k = Xw_k.$$

3. **Deflate the Predictors**:
   - After extracting t_k, the predictors are deflated to remove the information explained by t_k:

$$X \leftarrow X - t_k p_k',$$

   where $p_k = \frac{X't_k}{t_k't_k}$ are the loadings for X.
4. **Deflate the Response**:
   - Similarly, deflate y to remove the variance explained by t_k:

$$y \leftarrow y - t_k q_k,$$

   where $q_k = \frac{t_k'y}{t_k't_k}$.
5. **Repeat for All Components**:
   - Repeat the steps above until K components are extracted.

After constructing K components, the response y is modeled as:

$$y = TC + \epsilon,$$

where:

- $T = [t_1, t_2, \dots, t_K]$ is the matrix of latent components,
- C is the vector of regression coefficients.

The estimated coefficients for the original predictors are then:

$$\hat{\beta} = W(P'W)^{-1}C,$$

where $W = [w_1, w_2, \dots, w_K]$ and $P = [p_1, p_2, \dots, p_K]$.

5.6.3 Properties of PLS

1. **Dimensionality Reduction**:
   - PLS reduces X to K components, where $K \leq \min(n, p)$.
2. **Handles Multicollinearity**:
   - By constructing uncorrelated components, PLS avoids the instability caused by multicollinearity in OLS.
3. **Supervised Dimensionality Reduction**:
   - Unlike PCR, PLS considers the relationship between X and y when constructing components.
4. **Efficiency**:
   - PLS requires fewer components than PCR to achieve a similar level of predictive accuracy.

Practical Considerations

1. **Number of Components**:
   - The optimal number of components K can be determined using cross-validation.
2. **Preprocessing**:
   - Standardizing predictors is essential for PLS, as it ensures that all variables are on the same scale.
3. **Comparison with Other Methods**:
   - PLS outperforms OLS and PCR in cases of multicollinearity or when $p > n$, but it may be less interpretable than sparse methods like Lasso.

```r
# Load required library
library(pls)

# Step 1: Simulate data
set.seed(123)  # Ensure reproducibility
n <- 100       # Number of observations
p <- 10        # Number of predictors
X <- matrix(rnorm(n * p), nrow = n, ncol = p)  # Design matrix (predictors)
beta <- runif(p)                               # True coefficients
```

```r
y <- X %*% beta + rnorm(n)                    # Response variable with noise

# Step 2: Fit Partial Least Squares (PLS) Regression
pls_fit <- plsr(y ~ X, ncomp = 5, validation = "CV")

# Step 3: Summarize the PLS Model
summary(pls_fit)
#> Data:    X dimension: 100 10
#>  Y dimension: 100 1
#> Fit method: kernelpls
#> Number of components considered: 5
#>
#> VALIDATION: RMSEP
#> Cross-validated using 10 random segments.
#>        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps
#> CV           1.339    1.123    1.086    1.090    1.088    1.087
#> adjCV        1.339    1.112    1.078    1.082    1.080    1.080
#>
#> TRAINING: % variance explained
#>    1 comps  2 comps  3 comps  4 comps  5 comps
#> X    10.88    20.06    30.80    42.19    51.61
#> y    44.80    48.44    48.76    48.78    48.78

# Step 4: Perform Cross-Validation and Select Optimal Components
validationplot(pls_fit, val.type = "MSEP")
```

```r
# Step 5: Extract Coefficients for Predictors
pls_coefficients <- coef(pls_fit)
print(pls_coefficients)
#> , , 5 comps
#>
#>               y
#> X1   0.30192935
#> X2  -0.03161151
#> X3   0.22392538
#> X4   0.42315637
#> X5   0.33000198
#> X6   0.66228763
#> X7   0.40452691
#> X8  -0.05704037
#> X9  -0.02699757
#> X10  0.05944765

# Step 6: Evaluate Model Performance
predicted_y <- predict(pls_fit, X)
actual_vs_predicted <- data.frame(
  Actual = y,
  Predicted = predicted_y[, , 5]  # Predicted values using 5 components
)

# Plot Actual vs Predicted
library(ggplot2)
ggplot(actual_vs_predicted, aes(x = Actual, y = Predicted)) +
    geom_point() +
    geom_abline(
        intercept = 0,
        slope = 1,
        color = "red",
        linetype = "dashed"
    ) +
    labs(title = "Actual vs Predicted Values (PLS Regression)",
         x = "Actual Values",
         y = "Predicted Values") +
    theme_minimal()
```

```
# Step 7: Extract and Interpret Variable Importance (Loadings)
loadings_matrix <- as.matrix(unclass(loadings(pls_fit)))
variable_importance <- as.data.frame(loadings_matrix)
colnames(variable_importance) <-
    paste0("Component_", 1:ncol(variable_importance))
rownames(variable_importance) <-
    paste0("X", 1:nrow(variable_importance))

# Print variable importance
print(variable_importance)
#>     Component_1 Component_2 Component_3 Component_4 Component_5
#> X1  -0.04991097   0.5774569  0.24349681 -0.41550345 -0.02098351
#> X2   0.08913192  -0.1139342 -0.17582957 -0.05709948 -0.06707863
#> X3   0.13773357   0.1633338  0.07622919 -0.07248620 -0.61962875
#> X4   0.40369572  -0.2730457  0.69994206 -0.07949013  0.35239113
#> X5   0.50562681  -0.1788131 -0.27936562  0.36197480 -0.41919645
#> X6   0.57044281   0.3358522 -0.38683260  0.17656349  0.31154275
#> X7   0.36258623   0.1202109 -0.01753715 -0.12980483 -0.06919411
#> X8   0.12975452  -0.1164935 -0.30479310 -0.65654861  0.49948167
#> X9  -0.29521786   0.6170234 -0.32082508 -0.01041860  0.04904396
#> X10  0.23930055  -0.3259554  0.20006888 -0.53547258 -0.17963372
```

The loadings provide the contribution of each predictor to the PLS components. Higher absolute values indicate stronger contributions to the corresponding component.

- **Summary of the Model**:

  - The **proportion of variance explained** indicates how much of the variability in both the predictors and response is captured by each PLS component.
  - The goal is to retain enough components to explain most of the variance while avoiding overfitting.
- **Validation Plot**:
  - The **Mean Squared Error of Prediction (MSEP)** curve is used to select the optimal number of components.
  - Adding too many components can lead to overfitting, while too few may underfit the data.
- **Coefficients**:
  - The extracted coefficients are the weights applied to the predictors in the final PLS model.
  - These coefficients are derived from the PLS components and may differ from OLS regression coefficients due to dimensionality reduction.
- **Actual vs Predicted Plot**:
  - This visualization evaluates how well the PLS model predicts the response variable.
  - Points tightly clustered around the diagonal indicate good performance.
- **VIP Scores**:
  - VIP scores help identify the most important predictors in the PLS model.
  - Predictors with higher VIP scores contribute more to explaining the response variable.

5.6.4 Comparison with Related Methods

Method	Handles Multi-collinearity	Supervised Dimensionality Reduction	Sparse Solution	Interpretability
OLS	No	No	No	High
Ridge Regression	Yes	No	No	Moderate
Lasso Regression	Yes	No	Yes	High

Method	**Handles Multi-collinearity**	**Supervised Dimensionality Reduction**	**Sparse Solution**	**Interpretability**
PCR	Yes	No	No	Low
PLS	Yes	Yes	No	Moderate

Chapter 6

Non-Linear Regression

Non-linear regression models differ fundamentally from linear regression models in that the derivatives of the mean function with respect to parameters depend on one or more of the parameters. This dependence adds complexity but also provides greater flexibility to model intricate relationships.

Linear Regression:

- **Model Form Example:** A typical linear regression model looks like $y = \beta_0 + \beta_1 x$, where β_0 and β_1 are the parameters.
- **Parameter Effect:** The influence of each parameter on y is constant. For example, if β_1 increases by 1, the change in y is always x, regardless of the current value of β_1.
- **Derivatives:** The partial derivatives of y with respect to each parameter (e.g., $\frac{\partial y}{\partial \beta_1} = x$) do not depend on the parameters β_0 or β_1 themselves—they only depend on the data x. This makes the mathematics of finding the best-fit line straightforward.
- Straightforward estimation via closed-form solutions like Ordinary Least Squares.

Non-linear Regression:

- **Model Form Example:** Consider $y = \alpha \cdot e^{\beta x}$. Here, α and β are parameters, but the relationship is not a straight line.
- **Parameter Effect:** The effect of changing α or β on y is not constant. For instance, if you change β, the impact on y depends on both x and the current value of β. This makes predictions and adjustments more complex.
- **Derivatives:** Taking the partial derivative with respect to β gives $\frac{\partial y}{\partial \beta} =$

$\alpha x e^{\beta x}$. Notice this derivative depends on α, β, and x. Unlike linear regression, the sensitivity of y to changes in β changes as β itself changes.

- Estimation requires iterative algorithms like the Gauss-Newton Algorithm, as closed-form solutions are not feasible.

Table 6.1: Summary Table: Linear vs. Non-Linear Regression

Feature	Linear Regression	Non-Linear Regression
Relationship	Linear in parameters	Non-linear in parameters
Interpretability	High	Often challenging
Estimation	Closed-form solutions	Iterative algorithms
Computational Cost	Low	Higher

Key Features of Non-linear regression:

- **Complex Functional Forms**: Non-linear regression allows for relationships that are not straight lines or planes.
- **Interpretability Challenges**: Non-linear models can be difficult to interpret, especially if the functional forms are complex.
- **Practical Use Cases**:
  - Growth curves
  - High-order polynomials
  - Linear approximations (e.g., Taylor expansions)
  - Collections of locally linear models or basis functions (e.g., splines)

While these approaches can approximate data, they may suffer from interpretability issues or may not generalize well when data is sparse. Hence, intrinsically non-linear models are often preferred.

Intrinsically Non-Linear Models

The general form of an intrinsically non-linear regression model is:

$$Y_i = f(\mathbf{x}_i;\ ) + \epsilon_i$$

Where:

- $f(\mathbf{x}_i;\ )$: A **non-linear function** that relates $E(Y_i)$ to the independent variables $\mathbf{x}_i$.
- $\mathbf{x}_i$: A $k \times 1$ vector of independent variables (fixed).
- $\ :$ A $p \times 1$ vector of parameters.
- ϵ_i: Independent and identically distributed random errors, often assumed to have a mean of 0 and a constant variance σ^2. In some cases, $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.

Example: Exponential Growth Model

A common non-linear model is the exponential growth function:

$$y = \theta_1 e^{\theta_2 x} + \epsilon$$

Where:

- θ_1: Initial value.
- θ_2: Growth rate.
- x: Independent variable (e.g., time).
- ϵ: Random error.

6.1 Inference

Since $Y_i = f(\mathbf{x}_i, \theta) + \epsilon_i$, where $\epsilon_i \sim \text{iid}(0, \sigma^2)$, we can estimate parameters $(\hat{\theta})$ by minimizing the sum of squared errors:

$$\sum_{i=1}^{n} \left(Y_i - f(\mathbf{x}_i, \theta)\right)^2$$

Let $\hat{\theta}$ be the minimizer, the variance of residuals is estimated as:

$$s^2 = \hat{\sigma}_\epsilon^2 = \frac{\sum_{i=1}^{n} \left(Y_i - f(\mathbf{x}_i, \hat{\theta})\right)^2}{n - p}$$

where p is the number of parameters in , and n is the number of observations.

Asymptotic Distribution of $\hat{\theta}$

Under regularity conditions—most notably that $\epsilon_i \sim N(0, \sigma^2)$ or that n is sufficiently large for a central-limit-type argument—the parameter estimates $\hat{\theta}$ have the following asymptotic normal distribution:

$$\hat{\theta} \sim AN(\ , \sigma^2[\mathbf{F}(\theta)'\mathbf{F}(\theta)]^{-1})$$

where

- AN stands for "asymptotic normality."
- $\mathbf{F}(\theta)$ is the $n \times p$ Jacobian matrix of partial derivatives of $f(\mathbf{x}_i, \theta)$ with respect to , evaluated at $\hat{\theta}$. Specifically,

$$\mathbf{F}(\theta) = \begin{pmatrix} \frac{\partial f(\mathbf{x}_1,\theta)}{\partial \theta_1} & \cdots & \frac{\partial f(\mathbf{x}_1,\theta)}{\partial \theta_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(\mathbf{x}_n,\theta)}{\partial \theta_1} & \cdots & \frac{\partial f(\mathbf{x}_n,\theta)}{\partial \theta_p} \end{pmatrix}$$

Asymptotic normality means that as the sample size n becomes large, the sampling distribution of $\hat{\theta}$ approaches a normal distribution, which enables inference on the parameters.

6.1.1 Linear Functions of the Parameters

A "linear function of the parameters" refers to a quantity that can be written as $\mathbf{a}'\theta$, where $\mathbf{a}$ is some (constant) contrast vector. Common examples include:

- A single parameter θ_j (using a vector $\mathbf{a}$ with 1 in the j-th position and 0 elsewhere).
- Differences, sums, or other contrasts, e.g. $\theta_1 - \theta_2$.

Suppose we are interested in a linear combination of the parameters, such as $\theta_1 - \theta_2$. Define the contrast vector $\mathbf{a}$ as:

$$\mathbf{a} = (0, 1, -1)'$$

We then consider inference for $\mathbf{a}'$ ($\mathbf{a}$ can be p-dimensional vector). Using rules for the expectation and variance of a linear combination of a random vector $\mathbf{Z}$:

$$E(\mathbf{a}'\mathbf{Z}) = \mathbf{a}'E(\mathbf{Z})$$
$$\mathrm{Var}(\mathbf{a}'\mathbf{Z}) = \mathbf{a}'\mathrm{Var}(\mathbf{Z})\mathbf{a}$$

We have

$$E(\mathbf{a}\hat{}\,) = \mathbf{a}'E(\hat{\theta}) \approx \mathbf{a}'\theta$$
$$\mathrm{Var}(\mathbf{a}'\hat{\theta}) = \mathbf{a}'\mathrm{Var}(\hat{\theta})\mathbf{a} \approx \sigma^2\mathbf{a}'[\mathbf{F}(\ )'\mathbf{F}(\ )]^{-\mathbf{1}}\mathbf{a}$$

Hence,

$$\mathbf{a}\hat{}\,\sim AN(\mathbf{a}'\ , \sigma^2\mathbf{a}'[\mathbf{F}(\ )'\mathbf{F}(\ )]^{-\mathbf{1}}\mathbf{a})$$

Confidence Intervals for Linear Contrasts

Since $\mathbf{a}\hat{}$ is asymptotically independent of s^2 (up to order $O1/n$), a two-sided $100(1-\alpha)\%$ confidence interval for $\mathbf{a}'$ is given by:

$$\mathbf{a}' \ \pm t_{(1-\alpha/2,n-p)} s \sqrt{\mathbf{a}'[\mathbf{F}(\hat{\ })'\mathbf{F}(\hat{\ })]^{-1}\mathbf{a}}$$

where

- $t_{(1-\alpha/2,n-p)}$ is the critical value of the t-distribution with $n-p$ degrees of freedom.
- $s = \sqrt{\widehat{\sigma^2}_\epsilon}$ is the estimated standard deviation of residuals.

Special Case: A Single Parameter θ_j

If we focus on a single parameter θ_j, let $\mathbf{a}' = (0, \dots, 1, \dots, 0)$ (with 1 at the j-th position). Then, the confidence interval for θ_j becomes:

$$\hat{\theta}_j \pm t_{(1-\alpha/2,n-p)} s \sqrt{\widehat{c}^j}$$

where $\widehat{c}^j$ is the j-th diagonal element of $[\mathbf{F}(\hat{\theta})'\mathbf{F}(\hat{\theta})]^{-1}$.

6.1.2 Nonlinear Functions of Parameters

In many cases, we are interested in nonlinear functions of θ. Let $h(\theta)$ be such a function (e.g., a ratio of parameters, a difference of exponentials, etc.).

When $h(\theta)$ is a nonlinear function of the parameters, we can use a **Taylor series expansion** about θ to approximate $h(\hat{\theta})$:

$$h(\hat{\theta}) \approx h(\theta) + \mathbf{h}'[\hat{\theta} - \theta]$$

where

- $\mathbf{h} = \left(\frac{\partial h}{\partial \theta_1}, \frac{\partial h}{\partial \theta_2}, \dots, \frac{\partial h}{\partial \theta_p}\right)'$ is the gradient vector of partial derivatives.

Key Approximations:

1. **Expectation and Variance of $\hat{\theta}$** (using the asymptotic normality of $\hat{\theta}$:

$$\begin{aligned} E(\hat{\theta}) &\approx \theta, \\ \mathrm{Var}(\hat{\theta}) &\approx \sigma^2[\mathbf{F}(\theta)'\mathbf{F}(\theta)]^{-1}. \end{aligned}$$

2. **Expectation and Variance of $h(\hat{\theta})$** (properties of expectation and variance of (approximately) linear transformations):

$$\begin{aligned} E(h(\hat{\theta})) &\approx h(\theta), \\ \mathrm{Var}(h(\hat{\theta})) &\approx \sigma^2\mathbf{h}'[\mathbf{F}(\theta)'\mathbf{F}(\theta)]^{-1}\mathbf{h}. \end{aligned}$$

Combining these results, we find:

$$h(\hat{\theta}) \sim AN(h(\theta), \sigma^2 \mathbf{h}'[\mathbf{F}(\theta)'\mathbf{F}(\theta)]^{-1}\mathbf{h}),$$

where AN represents asymptotic normality.

Confidence Interval for $h(\theta)$:

An approximate $100(1-\alpha)\%$ confidence interval for $h(\theta)$ is:

$$h(\hat{\theta}) \pm t_{(1-\alpha/2, n-p)} s \sqrt{\mathbf{h}'[\mathbf{F}(\theta)'\mathbf{F}(\theta)]^{-1}\mathbf{h}},$$

where $\mathbf{h}$ and $\mathbf{F}(\theta)$ are evaluated at $\hat{\theta}$.

To compute a **prediction interval** for a new observation Y_0 at $x = x_0$:

1. **Model Definition**:

$$Y_0 = f(x_0; \theta) + \epsilon_0, \quad \epsilon_0 \sim N(0, \sigma^2),$$

with the predicted value:

$$\hat{Y}_0 = f(x_0, \hat{\theta}).$$

2. **Approximation for $\hat{Y}_0$**: As $n \to \infty$, $\hat{\theta} \to \theta$, so we have:

$$f(x_0, \hat{\theta}) \approx f(x_0, \theta) + \mathbf{f}_0(\theta)'[\hat{\theta} - \theta],$$

where:

$$\mathbf{f}_0(\theta) = \left(\frac{\partial f(x_0, \theta)}{\partial \theta_1}, \dots, \frac{\partial f(x_0, \theta)}{\partial \theta_p} \right)'.$$

3. **Error Approximation**:

$$\begin{aligned} Y_0 - \hat{Y}_0 &\approx Y_0 - f(x_0, \theta) - f_0(\theta)'[\hat{\theta} - \theta] \\ &= \epsilon_0 - f_0(\theta)'[\hat{\theta} - \theta] \end{aligned}$$

4. **Variance of $Y_0 - \hat{Y}_0$**:

$$\begin{aligned} \mathrm{Var}(Y_0 - \hat{Y}_0) &\approx \mathrm{Var}(\epsilon_0 - \mathbf{f}_0(\theta)'[\hat{\theta} - \theta]) \\ &= \sigma^2 + \sigma^2 \mathbf{f}_0(\theta)'[\mathbf{F}(\theta)'\mathbf{F}(\theta)]^{-1}\mathbf{f}_0(\theta) \\ &= \sigma^2(1 + \mathbf{f}_0(\theta)'[\mathbf{F}(\theta)'\mathbf{F}(\theta)]^{-1}\mathbf{f}_0(\theta)). \end{aligned}$$

Hence, the prediction error $Y_0 - \hat{Y}_0$ follows an asymptotic normal distribution:

$$Y_0 - \hat{Y}_0 \sim AN(0, \sigma^2(1 + \mathbf{f}_0(\theta)'[\mathbf{F}(\theta)'\mathbf{F}(\theta)]^{-1}\mathbf{f}_0(\theta))).$$

A $100(1-\alpha)\%$ prediction interval for Y_0 is:

$$\hat{Y}_0 \pm t_{(1-\alpha/2, n-p)} s \sqrt{1 + \mathbf{f}_0(\hat{\theta})'[\mathbf{F}(\hat{\theta})'\mathbf{F}(\hat{\theta})]^{-1}\mathbf{f}_0(\hat{\theta})}.$$

where we substitute $\hat{\theta}$ into $\mathbf{f}_0$ and $\mathbf{F}$. Recall s is the estiamte of σ.

Sometimes we want a confidence interval for $E(Y_i)$ (i.e., the mean response at x_0), rather than a prediction interval for an individual future observation. In that case, the variance term for the random error ϵ_0 is not included. Hence, the formula is the same but without the "+1":

$$E(Y_0) \approx f(x_0; \theta),$$

and the confidence interval is:

$$f(x_0, \hat{\theta}) \pm t_{(1-\alpha/2, n-p)} s \sqrt{\mathbf{f}_0(\hat{\theta})'[\mathbf{F}(\hat{\theta})'\mathbf{F}(\hat{\theta})]^{-1}\mathbf{f}_0(\hat{\theta})}.$$

Summary

- Linear Functions of the Parameters: A function $f(x, \theta)$ is linear in θ if it can be written in the form

  $$f(x, \theta) = \theta_1 g_1(x) + \theta_2 g_2(x) + \cdots + \theta_p g_p(x)$$

  where $g_j(x)$ do not depend on θ. In this case, the Jacobian $\mathbf{F}(\theta)$ does not depend on θ itself (only on x_i), and exact formulas often match what is familiar from linear regression.

- Nonlinear Functions of Parameters: If $f(x, \theta)$ depends on θ in a nonlinear way (e.g., $\theta_1 e^{\theta_2 x}, \beta_1/\beta_2$ or more complicated expressions), $\mathbf{F}(\theta)$ depends on θ. Estimation generally requires iterative numerical methods (e.g., Gauss–Newton, Levenberg–Marquardt), and the asymptotic results rely on evaluating partial derivatives at $\hat{\theta}$. Nevertheless, the final inference formulas—confidence intervals, prediction intervals—have a similar form, thanks to the asymptotic normality arguments.

6.2 Non-linear Least Squares Estimation

The least squares (LS) estimate of θ, denoted as $\hat{\theta}$, minimizes the residual sum of squares:

$$S(\hat{\theta}) = SSE(\hat{\theta}) = \sum_{i=1}^{n} \{Y_i - f(\mathbf{x}_i; \hat{\theta})\}^2$$

To solve this, we consider the partial derivatives of $S(\theta)$ with respect to each θ_j and set them to zero, leading to the normal equations:

$$\frac{\partial S(\theta)}{\partial \theta_j} = -2 \sum_{i=1}^{n} \{Y_i - f(\mathbf{x}_i; \theta)\} \frac{\partial f(\mathbf{x}_i; \theta)}{\partial \theta_j} = 0$$

However, these equations are inherently non-linear and, in most cases, cannot be solved analytically. As a result, various estimation techniques are employed to approximate solutions efficiently. These approaches include:

- Iterative Optimization – Methods that refine estimates through successive iterations to minimize error.
- Derivative-Free Methods – Techniques that do not rely on gradient information, useful for complex or non-smooth functions.
- Stochastic Heuristic – Algorithms that incorporate randomness, such as genetic algorithms or simulated annealing, to explore solution spaces.
- Linearization– Approximating non-linear models with linear ones to enable analytical or numerical solutions.
- Hybrid Approaches – Combining multiple methods to leverage their respective strengths for improved estimation.

Category	**Method**	**Best For**	**Derivative**	**Optimization**	**Comp. Cost**
Iterative Optimization	Steepest Descent (Gradient Descent)	Simple problems, slow convergence	Yes	Local	Low
Iterative Optimization	Gauss-Newton Algorithm	Faster than GD, ignores exact second-order info	Yes	Local	Medium

Category	Method	Best For	Derivatives	Optimization	Comp. Cost
Iterative Optimization	Levenberg-Marquardt Algorithm	Balances GN & GD, robust	Yes	Local	Medium
Iterative Optimization	Newton-Raphson Method	Quadratic convergence, needs Hessian	Yes	Local	High
Iterative Optimization	Quasi-Newton Method	Approximates Hessian for large problems	Yes	Local	Medium
Iterative Optimization	Trust-Region Reflective Algorithm	Handles constraints, robust	Yes	Local	High
Derivative-Free	Secant Method	Approximates derivative from function evaluations	No	Local	Medium
Derivative-Free	Nelder-Mead (Simplex)	No derivatives, heuristic	No	Local	Medium
Derivative-Free	Powell's Method	Line search, no explicit gradient	No	Local	Medium
Derivative-Free	Grid Search	Exhaustive search (best in low dims)	No	Global	Very High

Category	Method	Best For	Derivative?	Optimization	Comp. Cost
Derivative-Free	Hooke-Jeeves Pattern Search	Pattern-based, black-box optimization	No	Local	Medium
Derivative-Free	Bisection Method	Simple root/interval-based approach	No	Local	Low
Stochastic Heuristic	Simulated Annealing	Escapes local minima, non-smooth problems	No	Global	High
Stochastic Heuristic	Genetic Algorithm	Large search spaces, evolving parameters	No	Global	High
Stochastic Heuristic	Particle Swarm Optimization	Swarm-based, often fast global convergence	No	Global	Medium
Stochastic Heuristic	Evolutionary Strategies	High-dimensional, adaptive step-size	No	Global	High
Stochastic Heuristic	Differential Evolution Algorithm	Robust global optimizer, population-based	No	Global	High

Category	Method	Best For	Derivative	Optimization	Comp. Cost
Stochastic Heuristic	Ant Colony Optimization (ACO)	Discrete / combinatorial problems	No	Global	High
Linearization	Taylor Series Approximation	Local approximation of non-linearity	Yes	Local	Low
Linearization	Log-Linearization	Transforms non-linear equations	Yes	Local	Low
Hybrid	Adaptive Levenberg-Marquardt	Dynamically adjusts damping in LM	Yes	Local	Medium
Hybrid	Hybrid Genetic Algorithm & LM (GA-LM)	GA for coarse search, LM for fine-tuning	No	Hybrid	High
Hybrid	Neural Network-Based NLLS	Deep learning for complex non-linear least squares	No	Hybrid	Very High

6.2.1 Iterative Optimization

6.2.1.1 Gauss-Newton Algorithm

The Gauss-Newton Algorithm is an iterative optimization method used to estimate parameters in nonlinear least squares problems. It refines parameter estimates by approximating the Hessian matrix using first-order derivatives, making it computationally efficient for many practical applications (e.g., regression models in finance and marketing analytics). The objective is to minimize the Sum

of Squared Errors (SSE):

$$SSE(\theta) = \sum_{i=1}^{n} [Y_i - f_i(\theta)]^2,$$

where $\mathbf{Y} = [Y_1, \dots, Y_n]'$ are the observed responses, and $f_i(\theta)$ are the model-predicted values.

Iterative Refinement via Taylor Expansion

The Gauss-Newton algorithm iteratively refines an initial estimate $\widehat{\theta}^{(0)}$ using the **Taylor series expansion** of $f(\mathbf{x}_i; \theta)$ about $\widehat{\theta}^{(0)}$. We start with the observation model:

$$Y_i = f(\mathbf{x}_i; \theta) + \epsilon_i.$$

By expanding $f(\mathbf{x}_i; \theta)$ around $\widehat{\theta}^{(0)}$ and ignoring higher-order terms (assuming the remainder is small), we get:

$$Y_i \approx f(\mathbf{x}_i; \widehat{\theta}^{(0)}) + \sum_{j=1}^{p} \frac{\partial f(\mathbf{x}_i; \theta)}{\partial \theta_j}\bigg|_{\theta=\widehat{\theta}^{(0)}} (\theta_j - \widehat{\theta}_j^{(0)}) + \epsilon_i.$$

In **matrix form**, let

$$\mathbf{Y} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix}, \quad \mathbf{f}(\widehat{\theta}^{(0)}) = \begin{bmatrix} f(\mathbf{x}_1, \widehat{\theta}^{(0)}) \\ \vdots \\ f(\mathbf{x}_n, \widehat{\theta}^{(0)}) \end{bmatrix},$$

and define the **Jacobian matrix** of partial derivatives

$$\mathbf{F}(\widehat{\theta}^{(0)}) = \begin{bmatrix} \frac{\partial f(\mathbf{x}_1, \theta)}{\partial \theta_1} & \cdots & \frac{\partial f(\mathbf{x}_1, \theta)}{\partial \theta_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(\mathbf{x}_n, \theta)}{\partial \theta_1} & \cdots & \frac{\partial f(\mathbf{x}_n, \theta)}{\partial \theta_p} \end{bmatrix}_{\theta=\widehat{\theta}^{(0)}}.$$

Then,

$$\mathbf{Y} \approx \mathbf{f}(\widehat{\theta}^{(0)}) + \mathbf{F}(\widehat{\theta}^{(0)})\,(\theta - \widehat{\theta}^{(0)}) + \epsilon,$$

with $\epsilon = [\epsilon_1, \dots, \epsilon_n]'$ assumed i.i.d. with mean 0 and variance σ^2.

From this linear approximation,

$$\mathbf{Y} - \mathbf{f}(\widehat{\theta}^{(0)}) \approx \mathbf{F}(\widehat{\theta}^{(0)})\,(\theta - \widehat{\theta}^{(0)}).$$

Solving for $\theta - \widehat{\theta}^{(0)}$ in the **least squares** sense gives the Gauss increment $\widehat{\delta}^{(1)}$, so we can update:

$$\widehat{\theta}^{(1)} = \widehat{\theta}^{(0)} + \widehat{\delta}^{(1)}.$$

Step-by-Step Procedure

1. **Initialize**: Start with an initial estimate $\widehat{\theta}^{(0)}$ and set $j = 0$.

2. **Compute Taylor Expansion**: Calculate $\mathbf{f}(\widehat{\theta}^{(j)})$ and $\mathbf{F}(\widehat{\theta}^{(j)})$.

3. **Solve for Increment**: Treating $\mathbf{Y} - \mathbf{f}(\widehat{\theta}^{(j)}) \approx \mathbf{F}(\widehat{\theta}^{(j)})\,(\theta - \widehat{\theta}^{(j)})$ as a linear model, use Ordinary Least Squares to compute $\widehat{\delta}^{(j+1)}$.

4. **Update Parameters**: Set $\widehat{\theta}^{(j+1)} = \widehat{\theta}^{(j)} + \widehat{\delta}^{(j+1)}$.

5. **Check for Convergence**: If the convergence criteria are met (see below), stop; otherwise, return to Step 2.

6. **Estimate Variance**: After convergence, we assume $\epsilon \sim (\mathbf{0}, \sigma^2\mathbf{I})$. The variance σ^2 can be estimated by

$$\widehat{\sigma}^2 = \frac{1}{n-p}(\mathbf{Y} - \mathbf{f}(\mathbf{x}; \widehat{\theta}))'(\mathbf{Y} - \mathbf{f}(\mathbf{x}; \widehat{\theta})).$$

Convergence Criteria

Common criteria for deciding when to stop iterating include:

1. **Objective Function Change**:

$$\frac{\left|SSE(\widehat{\theta}^{(j+1)}) - SSE(\widehat{\theta}^{(j)})\right|}{SSE(\widehat{\theta}^{(j)})} < \gamma_1.$$

2. **Parameter Change**:

$$\left|\widehat{\theta}^{(j+1)} - \widehat{\theta}^{(j)}\right| < \gamma_2.$$

3. **Residual Projection Criterion**: The residuals satisfy convergence as defined in (Bates and Watts, 1981).

Another way to see the update step is by viewing the **necessary condition for a minimum**: the gradient of $SSE(\theta)$ with respect to θ should be zero. For

$$SSE(\theta) = \sum_{i=1}^{n} [Y_i - f_i(\theta)]^2,$$

the gradient is

$$\frac{\partial SSE(\theta)}{\partial \theta} = 2\,\mathbf{F}(\theta)'\big[\mathbf{Y} - \mathbf{f}(\theta)\big].$$

Using the Gauss-Newton update rule from iteration j to $j+1$:

$$\begin{aligned}
\widehat{\theta}^{(j+1)} &= \widehat{\theta}^{(j)} + \widehat{\delta}^{(j+1)} \\
&= \widehat{\theta}^{(j)} + \big[\mathbf{F}(\widehat{\theta}^{(j)})'\,\mathbf{F}(\widehat{\theta}^{(j)})\big]^{-1}\,\mathbf{F}(\widehat{\theta}^{(j)})'\big[\mathbf{Y} - \mathbf{f}(\widehat{\theta}^{(j)})\big] \\
&= \widehat{\theta}^{(j)} - \frac{1}{2}\big[\mathbf{F}(\widehat{\theta}^{(j)})'\,\mathbf{F}(\widehat{\theta}^{(j)})\big]^{-1}\,\frac{\partial SSE(\widehat{\theta}^{(j)})}{\partial \theta},
\end{aligned}$$

where:

- $\frac{\partial SSE(\widehat{\theta}^{(j)})}{\partial \theta}$ is the **gradient vector**, pointing in the direction of **steepest ascent** of SSE.
- $\big[\mathbf{F}(\widehat{\theta}^{(j)})'\mathbf{F}(\widehat{\theta}^{(j)})\big]^{-1}$ determines the **step size**, controlling how far to move in the direction of improvement.
- The factor $-\frac{1}{2}$ ensures movement in the **direction of steepest descent**, helping to **minimize** the SSE.

The Gauss-Newton method works well when the nonlinear model can be approximated accurately by a **first-order Taylor expansion** near the solution. If the assumption of near-linearity in the residual function $\mathbf{r}(\theta) = \mathbf{Y} - \mathbf{f}(\theta)$ is violated, convergence may be slow or fail altogether. In such cases, more robust methods like Levenberg-Marquardt Algorithm (which modifies Gauss-Newton with a damping parameter) are often preferred.

```r
# Load necessary libraries
library(minpack.lm)  # Provides nonlinear least squares functions

# Define a nonlinear function (exponential model)
nonlinear_model <- function(theta, x) {
    # theta is a vector of parameters: theta[1] = A, theta[2] = B
    # x is the independent variable
    # The model is A * exp(B * x)
```

```
    theta[1] * exp(theta[2] * x)
}

# Define SSE function for clarity
sse <- function(theta, x, y) {
    # SSE = sum of squared errors between actual y and model predictions
    sum((y - nonlinear_model(theta, x)) ^ 2)
}

# Generate synthetic data
set.seed(123)                      # for reproducibility
x <- seq(0, 10, length.out = 100) # 100 points from 0 to 10
true_theta <- c(2, 0.3)            # true parameter values
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.5)

# Display the first few data points
head(data.frame(x, y))
#>           x        y
#> 1 0.0000000 1.719762
#> 2 0.1010101 1.946445
#> 3 0.2020202 2.904315
#> 4 0.3030303 2.225593
#> 5 0.4040404 2.322373
#> 6 0.5050505 3.184724

# Gauss-Newton optimization using nls.lm (Levenberg-Marquardt as extension).
# Initial guess for theta: c(1, 0.1)
fit <- nls.lm(
    par = c(1, 0.1),
    fn = function(theta)
        y - nonlinear_model(theta, x)
)

# Display estimated parameters
cat("Estimated parameters (A, B):\n")
#> Estimated parameters (A, B):
print(fit$par)
#> [1] 1.9934188 0.3008742
```

1. We defined the model "nonlinear_model(theta, x)" which returns A*exp(B*x).
2. We generated synthetic data using the "true_theta" values and added random noise.
3. We used `nls.lm(...)` from the `minpack.lm` package to fit the data:
   - `par = c(1, 0.1)` is our initial parameter guess.

- `fn = function(theta) y - nonlinear_model(theta, x)` is the residual function, i.e., observed minus predicted.

4. The `fit$par` provides the estimated parameters after the algorithm converges.

```r
# Visualize the data and the fitted model
plot(
  x,
  y,
  main = "Data and Fitted Curve (Gauss-Newton/Levenberg-Marquardt)",
  xlab = "x",
  ylab = "y",
  pch = 19,
  cex = 0.5
)
curve(
  nonlinear_model(fit$par, x),
  from = 0,
  to = 10,
  add = TRUE,
  col = "red",
  lwd = 2
)
legend(
  "topleft",
  legend = c("Data", "Fitted Curve"),
  pch = c(19, NA),
  lty = c(NA, 1),
  col = c("black", "red")
)
```

6.2.1.2 Modified Gauss-Newton Algorithm

The Modified Gauss-Newton Algorithm introduces a learning rate α_j to control step size and prevent overshooting the local minimum. The standard Gauss-Newton Algorithm assumes that the full step direction $\hat{\delta}^{(j+1)}$ is optimal, but in practice, especially for highly nonlinear problems, it can overstep the minimum or cause numerical instability. The modification introduces a step size reduction, making it more robust.

We redefine the update step as:

$$\hat{\theta}^{(j+1)} = \hat{\theta}^{(j)} + \alpha_j \hat{\delta}^{(j+1)}, \quad 0 < \alpha_j < 1,$$

where:

- α_j is a **learning rate**, controlling how much of the step $\hat{\delta}^{(j+1)}$ is taken.
- If $\alpha_j = 1$, we recover the standard Gauss-Newton method.
- If α_j is too small, convergence is slow; if too large, the algorithm may diverge.

This learning rate α_j allows for **adaptive step size adjustments**, helping prevent excessive parameter jumps and ensuring that SSE decreases at each iteration.

A common approach to determine α_j is **step halving**, ensuring that each iteration moves in a direction that reduces SSE. Instead of using a fixed α_j, we iteratively reduce the step size until SSE decreases:

$$\hat{\theta}^{(j+1)} = \hat{\theta}^{(j)} + \frac{1}{2^k}\hat{\delta}^{(j+1)},$$

where:

- k is the smallest non-negative integer such that

$$SSE(\hat{\theta}^{(j)} + \frac{1}{2^k}\hat{\delta}^{(j+1)}) < SSE(\hat{\theta}^{(j)}).$$

This means we start with the full step $\hat{\delta}^{(j+1)}$, then try $\hat{\delta}^{(j+1)}/2$, then $\hat{\delta}^{(j+1)}/4$, and so on, until SSE is reduced.

Algorithm for Step Halving:

1. Compute the Gauss-Newton step $\hat{\delta}^{(j+1)}$.
2. Set an initial $\alpha_j = 1$.
3. If the updated parameters $\hat{\theta}^{(j)} + \alpha_j\hat{\delta}^{(j+1)}$ increase SSE, divide α_j by 2.
4. Repeat until SSE decreases.

This **ensures monotonic SSE reduction**, preventing divergence due to an overly aggressive step.

Generalized Form of the Modified Algorithm

A more general form of the update rule, incorporating step size control and a matrix $\mathbf{A}_j$, is:

$$\hat{\theta}^{(j+1)} = \hat{\theta}^{(j)} - \alpha_j\mathbf{A}_j\frac{\partial Q(\hat{\theta}^{(j)})}{\partial\theta},$$

where:

- $\mathbf{A}_j$ is a **positive definite matrix** that preconditions the update direction.
- α_j is the **learning rate**.
- $\frac{\partial Q(\hat{\theta}^{(j)})}{\partial\theta}$ is the **gradient** of the objective function $Q(\theta)$, typically SSE in nonlinear regression.

Connection to the Modified Gauss-Newton Algorithm

The **Modified Gauss-Newton Algorithm** fits this framework:

$$\hat{\theta}^{(j+1)} = \hat{\theta}^{(j)} - \alpha_j[\mathbf{F}(\hat{\theta}^{(j)})'\mathbf{F}(\hat{\theta}^{(j)})]^{-1}\frac{\partial SSE(\hat{\theta}^{(j)})}{\partial\theta}.$$

Here, we recognize:

- **Objective function**: $Q = SSE$.
- **Preconditioner matrix**: $[\mathbf{F}(\hat{\theta}^{(j)})'\mathbf{F}(\hat{\theta}^{(j)})]^{-1} = \mathbf{A}$.

Thus, the standard Gauss-Newton method can be interpreted as a special case of this broader optimization framework, with a preconditioned gradient descent approach.

```r
# Load required library
library(minpack.lm)

# Define a nonlinear function (exponential model)
nonlinear_model <- function(theta, x) {
  theta[1] * exp(theta[2] * x)
}

# Define the Sum of Squared Errors function
sse <- function(theta, x, y) {
  sum((y - nonlinear_model(theta, x)) ^ 2)
}

# Generate synthetic data
set.seed(123)
x <- seq(0, 10, length.out = 100)
true_theta <- c(2, 0.3)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.5)

# Gauss-Newton with Step Halving
gauss_newton_modified <-
  function(theta_init,
           x,
           y,
           tol = 1e-6,
           max_iter = 100) {
    theta <- theta_init
    for (j in 1:max_iter) {
      # Compute Jacobian matrix numerically
      epsilon <- 1e-6
      F_matrix <-
        matrix(0, nrow = length(y), ncol = length(theta))
      for (p in 1:length(theta)) {
        theta_perturb <- theta
        theta_perturb[p] <- theta[p] + epsilon
        F_matrix[, p] <-
          (nonlinear_model(theta_perturb, x)-nonlinear_model(theta, x))/epsilon
      }
```

```
      # Compute residuals
      residuals <- y - nonlinear_model(theta, x)

      # Compute Gauss-Newton step
      delta <-
        solve(t(F_matrix) %*% F_matrix) %*% t(F_matrix) %*% residuals

      # Step Halving Implementation
      alpha <- 1
      k <- 0
      while (sse(theta + alpha * delta, x, y) >= sse(theta, x, y) &&
             k < 10) {
        alpha <- alpha / 2
        k <- k + 1
      }

      # Update theta
      theta_new <- theta + alpha * delta

      # Check for convergence
      if (sum(abs(theta_new - theta)) < tol) {
        break
      }

      theta <- theta_new
    }
    return(theta)
  }

# Run Modified Gauss-Newton Algorithm
theta_init <- c(1, 0.1)  # Initial parameter guess
estimated_theta <- gauss_newton_modified(theta_init, x, y)

# Display estimated parameters
cat("Estimated parameters (A, B) with Modified Gauss-Newton:\n")
#> Estimated parameters (A, B) with Modified Gauss-Newton:
print(estimated_theta)
#>           [,1]
#> [1,] 1.9934188
#> [2,] 0.3008742

# Plot data and fitted curve
plot(
  x,
  y,
```

```
  main = "Modified Gauss-Newton: Data & Fitted Curve",
  pch = 19,
  cex = 0.5,
  xlab = "x",
  ylab = "y"
)
curve(
  nonlinear_model(estimated_theta, x),
  from = 0,
  to = 10,
  add = TRUE,
  col = "red",
  lwd = 2
)
legend(
  "topleft",
  legend = c("Data", "Fitted Curve"),
  pch = c(19, NA),
  lty = c(NA, 1),
  col = c("black", "red")
)
```

6.2.1.3 Steepest Descent (Gradient Descent)

The Steepest Descent Method, commonly known as Gradient Descent, is a fundamental iterative optimization technique used for finding parameter estimates that minimize an objective function $\mathbf{Q}(\theta)$. It is a special case of the Modified

Gauss-Newton Algorithm, where the preconditioning matrix $\mathbf{A}_j$ is replaced by the identity matrix.

The update rule is given by:

$$\widehat{\theta}^{(j+1)} = \widehat{\theta}^{(j)} - \alpha_j \mathbf{I}_{p\times p} \frac{\partial \mathbf{Q}(\widehat{\theta}^{(j)})}{\partial \theta},$$

where:

- α_j is the **learning rate**, determining the step size.
- $\mathbf{I}_{p\times p}$ is the **identity matrix**, meaning updates occur in the direction of the **negative gradient**.
- $\frac{\partial \mathbf{Q}(\widehat{\theta}^{(j)})}{\partial \theta}$ is the **gradient vector**, which provides the direction of **steepest ascent**; its negation ensures movement toward the minimum.

Characteristics of Steepest Descent

- **Slow to converge:** The algorithm moves in the direction of the gradient but does not take into account curvature, which may result in slow convergence, especially in ill-conditioned problems.
- **Moves rapidly initially:** The method can exhibit **fast initial progress**, but as it approaches the minimum, step sizes become small, leading to slow convergence.
- **Useful for initialization:** Due to its simplicity and ease of implementation, it is often used to obtain **starting values** for more advanced methods like **Newton's method** or Gauss-Newton Algorithm.

Comparison with Gauss-Newton

Method	Update Rule	Advantages	Disadvantages
Steepest Descent	$\widehat{\theta}^{(j+1)} = \widehat{\theta}^{(j)} - \alpha_j \mathbf{I} \nabla Q(\theta)$	Simple, requires only first derivatives	Slow convergence, sensitive to α_j
Gauss-Newton	$\widehat{\theta}^{(j+1)} = \widehat{\theta}^{(j)} - \mathbf{H}^{-1} \nabla Q(\theta)$	Uses curvature, faster near solution	Requires Jacobian computation, may diverge

The **key difference** is that Steepest Descent **only** considers the gradient direction, while Gauss-Newton and Newton's method incorporate curvature information.

Choosing the Learning Rate α_j

A well-chosen **learning rate** is crucial for the success of gradient descent:

- **Too large**: The algorithm may overshoot the minimum and diverge.
- **Too small**: Convergence is **very slow**.
- **Adaptive strategies**:
  - **Fixed step size**: α_j is constant.
  - **Step size decay**: α_j decreases over iterations (e.g., $\alpha_j = \frac{1}{j}$).
  - **Line search**: Choose α_j by minimizing $\mathbf{Q}(\theta^{(j+1)})$ along the gradient direction.

A common approach is **backtracking line search**, where α_j is reduced iteratively until a decrease in $\mathbf{Q}(\theta)$ is observed.

```r
# Load necessary libraries
library(ggplot2)


# Define the nonlinear function (exponential model)
nonlinear_model <- function(theta, x) {
    theta[1] * exp(theta[2] * x)
}

# Define the Sum of Squared Errors function
sse <- function(theta, x, y) {
    sum((y - nonlinear_model(theta, x)) ^ 2)
}

# Define Gradient of SSE w.r.t theta (computed numerically)
gradient_sse <- function(theta, x, y) {
    n <- length(y)
    residuals <- y - nonlinear_model(theta, x)

    # Partial derivative w.r.t theta_1
    grad_1 <- -2 * sum(residuals * exp(theta[2] * x))

    # Partial derivative w.r.t theta_2
    grad_2 <- -2 * sum(residuals * theta[1] * x * exp(theta[2] * x))

    return(c(grad_1, grad_2))
}

# Generate synthetic data
set.seed(123)
x <- seq(0, 10, length.out = 100)
```

```r
true_theta <- c(2, 0.3)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.5)

# Safe Gradient Descent Implementation
gradient_descent <-
    function(theta_init,
             x,
             y,
             alpha = 0.01,
             tol = 1e-6,
             max_iter = 500) {
        theta <- theta_init
        sse_values <- numeric(max_iter)

        for (j in 1:max_iter) {
            grad <- gradient_sse(theta, x, y)

            # Check for NaN or Inf values in the gradient (prevents divergence)
            if (any(is.na(grad)) || any(is.infinite(grad))) {
                cat("Numerical instability detected at iteration",
                    j,
                    "\n")
                break
            }

            # Update step
            theta_new <- theta - alpha * grad
            sse_values[j] <- sse(theta_new, x, y)

            # Check for convergence
            if (!is.finite(sse_values[j])) {
                cat("Divergence detected at iteration", j, "\n")
                break
            }

            if (sum(abs(theta_new - theta)) < tol) {
                cat("Converged in", j, "iterations.\n")
                return(list(theta = theta_new, sse_values = sse_values[1:j]))
            }

            theta <- theta_new
        }

        return(list(theta = theta, sse_values = sse_values))
    }
```

```r
# Run Gradient Descent with a Safe Implementation
theta_init <- c(1, 0.1)  # Initial guess
alpha <- 0.001           # Learning rate
result <- gradient_descent(theta_init, x, y, alpha)
#> Divergence detected at iteration 1

# Extract results
estimated_theta <- result$theta
sse_values <- result$sse_values

# Display estimated parameters
cat("Estimated parameters (A, B) using Gradient Descent:\n")
#> Estimated parameters (A, B) using Gradient Descent:
print(estimated_theta)
#> [1] 1.0 0.1

# Plot convergence of SSE over iterations
# Ensure sse_values has valid data
sse_df <- data.frame(
  Iteration = seq_along(sse_values),
  SSE = sse_values
)

# Generate improved plot using ggplot()
ggplot(sse_df, aes(x = Iteration, y = SSE)) +
  geom_line(color = "blue", linewidth = 1) +
  labs(
    title = "Gradient Descent Convergence",
    x = "Iteration",
    y = "SSE"
  ) +
  theme_minimal()
```

1. Steepest Descent (Gradient Descent) moves in the direction of steepest descent, which can lead to zigzagging behavior.
2. Slow convergence occurs when the curvature of the function varies significantly across dimensions.
3. Learning rate tuning is critical:
   - If too large, the algorithm diverges.
   - If too small, progress is very slow.
4. Useful for initialization: It is often used to get close to the optimal solution before switching to more advanced methods like Gauss-Newton Algorithm or Newton's method.

Several advanced techniques improve the performance of steepest descent:

- **Momentum Gradient Descent**: Adds a momentum term to smooth updates, reducing oscillations.
- **Adaptive Learning Rates**:
  - AdaGrad: Adjusts α_j per parameter based on historical gradients.
  - RMSprop: Uses a moving average of past gradients to scale updates.
  - Adam (Adaptive Moment Estimation): Combines momentum and adaptive learning rates.

In practice, Adam is widely used in machine learning and deep learning, while Newton-based methods (including Gauss-Newton) are preferred for nonlinear regression.

6.2.1.4 Levenberg-Marquardt Algorithm

The Levenberg-Marquardt Algorithm is a widely used optimization method for solving nonlinear least squares problems. It is an adaptive technique that blends the Gauss-Newton Algorithm and Steepest Descent (Gradient Descent), dynamically switching between them based on problem conditions.

The update rule is:

$$\hat{\theta}^{(j+1)} = \hat{\theta}^{(j)} - \alpha_j [\mathbf{F}(\hat{\theta}^{(j)})' \mathbf{F}(\hat{\theta}^{(j)}) + \tau \mathbf{I}_{p \times p}] \frac{\partial \mathbf{Q}(\hat{\theta}^{(j)})}{\partial \theta}$$

where:

- τ is the **damping parameter**, controlling whether the step behaves like Gauss-Newton Algorithm or Steepest Descent (Gradient Descent).
- $\mathbf{I}_{p \times p}$ is the **identity matrix**, ensuring numerical stability.
- $\mathbf{F}(\hat{\theta}^{(j)})$ is the **Jacobian matrix** of partial derivatives.
- $\frac{\partial \mathbf{Q}(\hat{\theta}^{(j)})}{\partial \theta}$ is the **gradient vector**.
- α_j is the **learning rate**, determining step size.

The Levenberg-Marquardt algorithm is particularly useful when the Jacobian matrix $\mathbf{F}(\hat{\theta}^{(j)})$ is nearly singular, meaning that Gauss-Newton Algorithm alone may fail.

- When τ is large, the method behaves like Steepest Descent, ensuring stability.
- When τ is small, it behaves like Gauss-Newton, accelerating convergence.
- Adaptive control of τ:
  - If $SSE(\hat{\theta}^{(j+1)}) < SSE(\hat{\theta}^{(j)})$, reduce τ:

$$\tau \leftarrow \tau/10$$

  - Otherwise, increase τ to stabilize:

$$\tau \leftarrow 10\tau$$

This adjustment ensures that the algorithm moves efficiently while avoiding instability.

```r
# Load required libraries
library(minpack.lm)
library(ggplot2)

# Define a nonlinear function (exponential model)
nonlinear_model <- function(theta, x) {
    theta[1] * exp(theta[2] * x)
}
```

```r
# Define SSE function
sse <- function(theta, x, y) {
    sum((y - nonlinear_model(theta, x)) ^ 2)
}

# Generate synthetic data
set.seed(123)
x <- seq(0, 10, length.out = 100)
true_theta <- c(2, 0.3)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.5)

# Robust Levenberg-Marquardt Optimization Implementation
levenberg_marquardt <-
    function(theta_init,
             x,
             y,
             tol = 1e-6,
             max_iter = 500,
             tau_init = 1) {
        theta <- theta_init
        tau <- tau_init
        lambda <- 1e-8  # Small regularization term
        sse_values <- numeric(max_iter)

        for (j in 1:max_iter) {
            # Compute Jacobian matrix numerically
            epsilon <- 1e-6
            F_matrix <-
                matrix(0, nrow = length(y), ncol = length(theta))
            for (p in 1:length(theta)) {
                theta_perturb <- theta
                theta_perturb[p] <- theta[p] + epsilon
                F_matrix[, p] <-
                    (nonlinear_model(theta_perturb, x)
                     - nonlinear_model(theta, x)) / epsilon
            }

            # Compute residuals
            residuals <- y - nonlinear_model(theta, x)

            # Compute Levenberg-Marquardt update
            A <-
                t(F_matrix) %*% F_matrix + tau * diag(length(theta))
            + lambda * diag(length(theta))  # Regularized A
            delta <- tryCatch(
```

```
                solve(A) %*% t(F_matrix) %*% residuals,
                error = function(e) {
                    cat("Singular matrix detected at iteration",
                        j,
                        "- Increasing tau\n")
                    tau <<- tau * 10  # Increase tau to stabilize
                    # Return zero delta to avoid NaN updates
                    return(rep(0, length(theta)))
                }
            )

            theta_new <- theta + delta

            # Compute new SSE
            sse_values[j] <- sse(theta_new, x, y)

            # Adjust tau dynamically
            if (sse_values[j] < sse(theta, x, y)) {
                # Reduce tau but prevent it from going too low
                tau <-
                    max(tau / 10, 1e-8)
            } else {
                tau <- tau * 10  # Increase tau if SSE increases
            }

            # Check for convergence
            if (sum(abs(delta)) < tol) {
                cat("Converged in", j, "iterations.\n")
                return(list(theta = theta_new, sse_values = sse_values[1:j]))
            }

            theta <- theta_new
        }

        return(list(theta = theta, sse_values = sse_values))
    }

# Run Levenberg-Marquardt
theta_init <- c(1, 0.1)  # Initial guess
result <- levenberg_marquardt(theta_init, x, y)
#> Singular matrix detected at iteration 11 - Increasing tau
#> Converged in 11 iterations.

# Extract results
estimated_theta <- result$theta
```

```r
sse_values <- result$sse_values

# Display estimated parameters
cat("Estimated parameters (A, B) using Levenberg-Marquardt:\n")
#> Estimated parameters (A, B) using Levenberg-Marquardt:
print(estimated_theta)
#>               [,1]
#> [1,] -6.473440e-09
#> [2,]  1.120637e+01

# Plot convergence of SSE over iterations
sse_df <-
    data.frame(Iteration = seq_along(sse_values), SSE = sse_values)

ggplot(sse_df, aes(x = Iteration, y = SSE)) +
    geom_line(color = "blue", linewidth = 1) +
    labs(title = "Levenberg-Marquardt Convergence",
         x = "Iteration",
         y = "SSE") +
    theme_minimal()
```

- Early Stability (Flat SSE)
  - The SSE remains near zero for the first few iterations, which suggests that the algorithm is initially behaving stably.
  - This might indicate that the initial parameter guess is reasonable, or that the updates are too small to significantly affect SSE.

- Sudden Explosion in SSE (Iteration ~8-9)
  - The sharp spike in SSE at iteration 9 indicates a numerical instability or divergence in the optimization process.
  - This could be due to:
    * An ill-conditioned Jacobian matrix: The step direction is poorly estimated, leading to an unstable jump.
    * A sudden large update (`delta`): The damping parameter (`tau`) might have been reduced too aggressively, causing an uncontrolled step.
    * Floating-point issues: If `A` becomes nearly singular, solving `A \ delta = residuals` may produce excessively large values.
- Return to Stability (After Iteration 9)
  - The SSE immediately returns to a low value after the spike, which suggests that the damping parameter (`tau`) might have been increased after detecting instability.
  - This is consistent with the adaptive nature of Levenberg-Marquardt:
    * If a step leads to a bad SSE increase, the algorithm increases `tau` to make the next step more conservative.
    * If the next step stabilizes, `tau` may be reduced again.

6.2.1.5 Newton-Raphson Algorithm

The Newton-Raphson method is a second-order optimization technique used for nonlinear least squares problems. Unlike first-order methods (such as Steepest Descent (Gradient Descent) and Gauss-Newton Algorithm), Newton-Raphson uses both first and second derivatives of the objective function for faster convergence.

The update rule is:

$$\hat{\theta}^{(j+1)} = \hat{\theta}^{(j)} - \alpha_j \left[\frac{\partial^2 Q(\hat{\theta}^{(j)})}{\partial\theta\partial\theta'} \right]^{-1} \frac{\partial Q(\hat{\theta}^{(j)})}{\partial\theta}$$

where:

- $\frac{\partial Q(\hat{\theta}^{(j)})}{\partial\theta}$ is the **gradient vector** (first derivative of the objective function).
- $\frac{\partial^2 Q(\hat{\theta}^{(j)})}{\partial\theta\partial\theta'}$ is the **Hessian matrix** (second derivative of the objective function).
- α_j is the **learning rate**, controlling step size.

The **Hessian matrix** in nonlinear least squares problems is:

$$\frac{\partial^2 Q(\hat{\theta}^{(j)})}{\partial\theta\partial\theta'} = 2\mathbf{F}(\hat{\theta}^{(j)})'\mathbf{F}(\hat{\theta}^{(j)}) - 2\sum_{i=1}^{n}[Y_i - f(x_i;\theta)]\frac{\partial^2 f(x_i;\theta)}{\partial\theta\partial\theta'}$$

where:

- The first term $2\mathbf{F}(\hat{\theta}^{(j)})'\mathbf{F}(\hat{\theta}^{(j)})$ is the same as in the Gauss-Newton Algorithm.
- The second term $-2\sum_{i=1}^{n}[Y_i - f(x_i;\theta)]\frac{\partial^2 f(x_i;\theta)}{\partial\theta\partial\theta'}$ contains second-order derivatives.

Key Observations

- Gauss-Newton vs. Newton-Raphson:
  - Gauss-Newton approximates the Hessian by ignoring the second term.
  - Newton-Raphson explicitly incorporates second-order derivatives, making it more precise but computationally expensive.
- Challenges:
  - The Hessian matrix may be singular, making it impossible to invert.
  - Computing second derivatives is often difficult for complex functions.

```r
# Load required libraries
library(ggplot2)

# Define a nonlinear function (exponential model)
nonlinear_model <- function(theta, x) {
    theta[1] * exp(theta[2] * x)
}

# Define SSE function
sse <- function(theta, x, y) {
    sum((y - nonlinear_model(theta, x)) ^ 2)
}

# Compute Gradient (First Derivative) of SSE
gradient_sse <- function(theta, x, y) {
    residuals <- y - nonlinear_model(theta, x)

    # Partial derivative w.r.t theta_1
    grad_1 <- -2 * sum(residuals * exp(theta[2] * x))

    # Partial derivative w.r.t theta_2
    grad_2 <- -2 * sum(residuals * theta[1] * x * exp(theta[2] * x))

    return(c(grad_1, grad_2))
```

```r
}

# Compute Hessian (Second Derivative) of SSE
hessian_sse <- function(theta, x, y) {
    residuals <- y - nonlinear_model(theta, x)

    # Compute second derivatives
    H_11 <- 2 * sum(exp(2 * theta[2] * x))
    H_12 <- 2 * sum(x * exp(2 * theta[2] * x) * theta[1])
    H_21 <- H_12

    term1 <- 2 * sum((x ^ 2) * exp(2 * theta[2] * x) * theta[1] ^ 2)
    term2 <- 2 * sum(residuals * (x ^ 2) * exp(theta[2] * x))

    H_22 <- term1 - term2

    return(matrix(
        c(H_11, H_12, H_21, H_22),
        nrow = 2,
        byrow = TRUE
    ))
}

# Generate synthetic data
set.seed(123)
x <- seq(0, 10, length.out = 100)
true_theta <- c(2, 0.3)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.5)

# Newton-Raphson Optimization Implementation
newton_raphson <-
    function(theta_init,
             x,
             y,
             tol = 1e-6,
             max_iter = 500) {
        theta <- theta_init
        sse_values <- numeric(max_iter)

        for (j in 1:max_iter) {
            grad <- gradient_sse(theta, x, y)
            hessian <- hessian_sse(theta, x, y)

            # Check if Hessian is invertible
            if (det(hessian) == 0) {
```

```
                cat("Hessian is singular at iteration",
                    j,
                    "- Using identity matrix instead.\n")
                # Replace with identity matrix if singular
                hessian <-
                    diag(length(theta))
            }

            # Compute Newton update
            delta <- solve(hessian) %*% grad
            theta_new <- theta - delta
            sse_values[j] <- sse(theta_new, x, y)

            # Check for convergence
            if (sum(abs(delta)) < tol) {
                cat("Converged in", j, "iterations.\n")
                return(list(theta = theta_new, sse_values = sse_values[1:j]))
            }

            theta <- theta_new
        }

        return(list(theta = theta, sse_values = sse_values))
    }

# Run Newton-Raphson
theta_init <- c(1, 0.1)  # Initial guess
result <- newton_raphson(theta_init, x, y)
#> Converged in 222 iterations.

# Extract results
estimated_theta <- result$theta
sse_values <- result$sse_values

# Display estimated parameters
cat("Estimated parameters (A, B) using Newton-Raphson:\n")
#> Estimated parameters (A, B) using Newton-Raphson:
print(estimated_theta)
#>           [,1]
#> [1,] 1.9934188
#> [2,] 0.3008742

# Plot convergence of SSE over iterations
sse_df <-
    data.frame(Iteration = seq_along(sse_values), SSE = sse_values)
```

```r
ggplot(sse_df, aes(x = Iteration, y = SSE)) +
    geom_line(color = "blue", size = 1) +
    labs(title = "Newton-Raphson Convergence",
         x = "Iteration",
         y = "SSE") +
    theme_minimal()
```

6.2.1.6 Quasi-Newton Method

The Quasi-Newton method is an optimization technique that approximates Newton's method without requiring explicit computation of the Hessian matrix. Instead, it iteratively constructs an approximation $\mathbf{H}_j$ to the Hessian based on the first derivative information.

The update rule is:

$$\hat{\theta}^{(j+1)} = \hat{\theta}^{(j)} - \alpha_j \mathbf{H}_j^{-1} \frac{\partial \mathbf{Q}(\hat{\theta}^{(j)})}{\partial \theta}$$

where:

- $\mathbf{H}_j$ is a **symmetric positive definite approximation** to the **Hessian matrix**.
- As $j \to \infty$, $\mathbf{H}_j$ gets closer to the **true Hessian**.
- $\frac{\partial Q(\hat{\theta}^{(j)})}{\partial \theta}$ is the **gradient vector**.
- α_j is the **learning rate**, controlling step size.

Why Use Quasi-Newton Instead of Newton-Raphson Method?

- Newton-Raphson requires computing the Hessian matrix explicitly, which is computationally expensive and may be singular.
- Quasi-Newton avoids computing the Hessian directly by approximating it iteratively.
- Among first-order methods (which only require gradients, not Hessians), Quasi-Newton methods perform best.

Hessian Approximation

Instead of directly computing the Hessian $\mathbf{H}_j$, Quasi-Newton methods update an approximation $\mathbf{H}_j$ iteratively.

One of the most widely used formulas is the **Broyden-Fletcher-Goldfarb-Shanno (BFGS) update**:

$$\mathbf{H}_{j+1} = \mathbf{H}_j + \frac{(\mathbf{s}_j \mathbf{s}_j')}{\mathbf{s}_j' \mathbf{y}_j} - \frac{\mathbf{H}_j \mathbf{y}_j \mathbf{y}_j' \mathbf{H}_j}{\mathbf{y}_j' \mathbf{H}_j \mathbf{y}_j}$$

where:

- $\mathbf{s}_j = \hat{\theta}^{(j+1)} - \hat{\theta}^{(j)}$ (change in parameters).
- $\mathbf{y}_j = \nabla Q(\hat{\theta}^{(j+1)}) - \nabla Q(\hat{\theta}^{(j)})$ (change in gradient).
- $\mathbf{H}_j$ is the current **inverse Hessian approximation**.

```r
# Load required libraries
library(ggplot2)

# Define a nonlinear function (exponential model)
nonlinear_model <- function(theta, x) {
    theta[1] * exp(theta[2] * x)
}

# Define SSE function
sse <- function(theta, x, y) {
    sum((y - nonlinear_model(theta, x))^2)
}

# Generate synthetic data
set.seed(123)
x <- seq(0, 10, length.out = 100)
true_theta <- c(2, 0.3)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.5)
```

```r
# Run BFGS Optimization using `optim()`
theta_init <- c(1, 0.1)  # Initial guess
result <- optim(
    par = theta_init,
    fn = function(theta) sse(theta, x, y),  # Minimize SSE
    method = "BFGS",
    control = list(trace = 0)  # Suppress optimization progress
    # control = list(trace = 1, REPORT = 1)  # Print optimization progress
)

# Extract results
estimated_theta <- result$par
sse_final <- result$value
convergence_status <- result$convergence  # 0 means successful convergence

# Display estimated parameters
cat("\n=== Optimization Results ===\n")
#>
#> === Optimization Results ===
cat("Estimated parameters (A, B) using Quasi-Newton BFGS:\n")
#> Estimated parameters (A, B) using Quasi-Newton BFGS:
print(estimated_theta)
#> [1] 1.9954216 0.3007569

# Display final SSE
cat("\nFinal SSE:", sse_final, "\n")
#>
#> Final SSE: 20.3227
```

6.2.1.7 Trust-Region Reflective Algorithm

The Trust-Region Reflective (TRR) algorithm is an optimization technique used for nonlinear least squares problems. Unlike Newton's method and gradient-based approaches, TRR dynamically restricts updates to a trust region, ensuring stability and preventing overshooting.

The goal is to minimize the objective function $Q(\theta)$ (e.g., Sum of Squared Errors, SSE):

$$\hat{\theta} = \arg\min_{\theta} Q(\theta)$$

Instead of taking a full Newton step, TRR solves the following **quadratic sub-problem**:

$$\min_{\delta} m_j(\delta) = Q(\hat{\theta}^{(j)}) + \nabla Q(\hat{\theta}^{(j)})'\delta + \frac{1}{2}\delta'\mathbf{H}_j\delta$$

subject to:

$$\|\delta\| \leq \Delta_j$$

where:

- $\mathbf{H}_j$ is an **approximation of the Hessian matrix**.
- $\nabla Q(\hat{\theta}^{(j)})$ is the **gradient vector**.
- Δ_j is the **trust-region radius**, which is adjusted dynamically.

Trust-Region Adjustments

The algorithm **modifies the step size dynamically** based on the ratio ρ_j:

$$\rho_j = \frac{Q(\hat{\theta}^{(j)}) - Q(\hat{\theta}^{(j)} + \delta)}{m_j(0) - m_j(\delta)}$$

- **If** $\rho_j > 0.75$ and $\|\delta\| = \Delta_j$, then expand the trust region:

$$\Delta_{j+1} = 2\Delta_j$$

- **If** $\rho_j < 0.25$, shrink the trust region:

$$\Delta_{j+1} = \frac{1}{2}\Delta_j$$

- **If** $\rho_j > 0$, accept the step; otherwise, reject it.

If a step **violates a constraint**, it is **reflected back** into the feasible region:

$$\hat{\theta}^{(j+1)} = \max(\hat{\theta}^{(j)} + \delta, \theta_{\min})$$

This ensures that the optimization **respects parameter bounds**.

```r
# Load required libraries
library(ggplot2)


# Define a nonlinear function (exponential model)
```

```r
nonlinear_model <- function(theta, x) {
    theta[1] * exp(theta[2] * x)
}

# Define SSE function
sse <- function(theta, x, y) {
    sum((y - nonlinear_model(theta, x)) ^ 2)
}

# Compute Gradient (First Derivative) of SSE
gradient_sse <- function(theta, x, y) {
    residuals <- y - nonlinear_model(theta, x)

    # Partial derivative w.r.t theta_1
    grad_1 <- -2 * sum(residuals * exp(theta[2] * x))

    # Partial derivative w.r.t theta_2
    grad_2 <- -2 * sum(residuals * theta[1] * x * exp(theta[2] * x))

    return(c(grad_1, grad_2))
}

# Compute Hessian Approximation of SSE
hessian_sse <- function(theta, x, y) {
    residuals <- y - nonlinear_model(theta, x)

    # Compute second derivatives
    H_11 <- 2 * sum(exp(2 * theta[2] * x))
    H_12 <- 2 * sum(x * exp(2 * theta[2] * x) * theta[1])
    H_21 <- H_12

    term1 <- 2 * sum((x ^ 2) * exp(2 * theta[2] * x) * theta[1] ^ 2)
    term2 <- 2 * sum(residuals * (x ^ 2) * exp(theta[2] * x))

    H_22 <- term1 - term2

    return(matrix(
        c(H_11, H_12, H_21, H_22),
        nrow = 2,
        byrow = TRUE
    ))
}

# Generate synthetic data
set.seed(123)
```

```r
x <- seq(0, 10, length.out = 100)
true_theta <- c(2, 0.3)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.5)

# Manual Trust-Region Reflective Optimization Implementation
trust_region_reflective <-
    function(theta_init,
             x,
             y,
             tol = 1e-6,
             max_iter = 500,
             delta_max = 1.0) {
        theta <- theta_init
        delta_j <- 0.5  # Initial trust-region size
        n <- length(theta)
        sse_values <- numeric(max_iter)

        for (j in 1:max_iter) {
            grad <- gradient_sse(theta, x, y)
            hessian <- hessian_sse(theta, x, y)

            # Check if Hessian is invertible
            if (det(hessian) == 0) {
                cat("Hessian is singular at iteration",
                    j,
                    "- Using identity matrix instead.\n")
                hessian <-
                    diag(n)  # Replace with identity matrix if singular
            }

            # Compute Newton step
            delta_full <- -solve(hessian) %*% grad

            # Apply trust-region constraint
            if (sqrt(sum(delta_full ^ 2)) > delta_j) {
                # Scale step
                delta <-
                    (delta_j / sqrt(sum(delta_full ^ 2))) * delta_full
            } else {
                delta <- delta_full
            }

            # Compute new theta and ensure it respects constraints
            theta_new <-
                pmax(theta + delta, c(0,-Inf))  # Reflect to lower bound
```

```
            sse_new <- sse(theta_new, x, y)

            # Compute agreement ratio (rho_j)
            predicted_reduction <-
                -t(grad) %*% delta - 0.5 * t(delta) %*% hessian %*% delta
            actual_reduction <- sse(theta, x, y) - sse_new
            rho_j <- actual_reduction / predicted_reduction

            # Adjust trust region size
            if (rho_j < 0.25) {
                delta_j <- max(delta_j / 2, 1e-4)  # Shrink
            } else if (rho_j > 0.75 &&
                       sqrt(sum(delta ^ 2)) == delta_j) {
                delta_j <- min(2 * delta_j, delta_max)  # Expand
            }

            # Accept or reject step
            if (rho_j > 0) {
                theta <- theta_new  # Accept step
            } else {
                cat("Step rejected at iteration", j, "\n")
            }

            sse_values[j] <- sse(theta, x, y)

            # Check for convergence
            if (sum(abs(delta)) < tol) {
                cat("Converged in", j, "iterations.\n")
                return(list(theta = theta, sse_values = sse_values[1:j]))
            }
        }

        return(list(theta = theta, sse_values = sse_values))
    }

# Run Manual Trust-Region Reflective Algorithm
theta_init <- c(1, 0.1)  # Initial guess
result <- trust_region_reflective(theta_init, x, y)

# Extract results
estimated_theta <- result$theta
sse_values <- result$sse_values

# Plot convergence of SSE over iterations
sse_df <-
```

```
    data.frame(Iteration = seq_along(sse_values), SSE = sse_values)

ggplot(sse_df, aes(x = Iteration, y = SSE)) +
    geom_line(color = "blue", size = 1) +
    labs(title = "Trust-Region Reflective Convergence",
         x = "Iteration",
         y = "SSE") +
    theme_minimal()
```

6.2.2 Derivative-Free

6.2.2.1 Secant Method

The Secant Method is a root-finding algorithm that approximates the derivative using finite differences, making it a derivative-free alternative to Newton's method. It is particularly useful when the exact gradient (or Jacobian in the case of optimization problems) is unavailable or expensive to compute.

In nonlinear optimization, we apply the Secant Method to iteratively refine parameter estimates without explicitly computing second-order derivatives.

In one dimension, the Secant Method approximates the derivative as:

$$f'(\theta) \approx \frac{f(\theta_j) - f(\theta_{j-1})}{\theta_j - \theta_{j-1}}.$$

Using this approximation, the **update step** in the Secant Method follows:

$$\theta_{j+1} = \theta_j - f(\theta_j)\frac{\theta_j - \theta_{j-1}}{f(\theta_j) - f(\theta_{j-1})}.$$

Instead of computing the exact derivative (as in Newton's method), we use the difference between the last two iterates to approximate it. This makes the Secant Method **more efficient** in cases where gradient computation is expensive or infeasible.

In **higher dimensions**, the Secant Method extends to an approximate Quasi-Newton Method, often referred to as **Broyden's Method**. We iteratively approximate the inverse Hessian matrix using past updates.

The update formula for a vector-valued function $F(\theta)$ is:

$$\theta^{(j+1)} = \theta^{(j)} - \mathbf{B}^{(j)} F(\theta^{(j)}),$$

where $\mathbf{B}^{(j)}$ is an approximation of the inverse Jacobian matrix, updated at each step using:

$$\mathbf{B}^{(j+1)} = \mathbf{B}^{(j)} + \frac{(\Delta\theta^{(j)} - \mathbf{B}^{(j)}\Delta F^{(j)})(\Delta\theta^{(j)})'}{(\Delta\theta^{(j)})'\Delta F^{(j)}},$$

where:

- $\Delta\theta^{(j)} = \theta^{(j+1)} - \theta^{(j)}$,
- $\Delta F^{(j)} = F(\theta^{(j+1)}) - F(\theta^{(j)})$.

This **secant-based update** approximates the behavior of the **true Jacobian inverse**, reducing computational cost compared to full Newton's method.

Algorithm: Secant Method for Nonlinear Optimization

The **Secant Method** for nonlinear optimization follows these steps:

1. Initialize parameters $\theta^{(0)}$ and $\theta^{(1)}$ (two starting points).
2. Compute the function values $F(\theta^{(0)})$ and $F(\theta^{(1)})$.
3. Use the Secant update formula to compute the next iterate $\theta^{(j+1)}$.
4. Update the approximate inverse Jacobian $\mathbf{B}^{(j)}$.
5. Repeat until convergence.

```r
# Load required libraries
library(numDeriv)


# Define a nonlinear function (logistic model)
nonlinear_model <- function(theta, x) {
    return(theta[1] / (1 + exp(-theta[2] * (x - theta[3]))))
}

# Define the Sum of Squared Errors (SSE) function
sse <- function(theta, x, y) {
    return(sum((y - nonlinear_model(theta, x)) ^ 2))
}

# Generate synthetic data
set.seed(123)
x <- seq(-5, 5, length.out = 100)
true_theta <- c(4, 1.5, 0)  # True parameters (A, B, C)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.3)

# Improved Secant Method with Line Search
secant_method_improved <-
    function(theta0,
             theta1,
             x,
             y,
             tol = 1e-6,
             max_iter = 100) {
        theta_prev <- as.matrix(theta0)  # Convert to column vector
        theta_curr <- as.matrix(theta1)

        alpha <- 1  # Initial step size
        step_reduction_factor <- 0.5  # Reduce step if SSE increases
        B_inv <- diag(length(theta0))  # Identity matrix initialization

        for (j in 1:max_iter) {
            # Compute function values using numerical gradient
            F_prev <-
                as.matrix(grad(function(theta)
                    sse(theta, x, y), theta_prev))
            F_curr <-
                as.matrix(grad(function(theta)
                    sse(theta, x, y), theta_curr))

            # Compute secant step update (convert to column vectors)
```

```r
        delta_theta <- as.matrix(theta_curr - theta_prev)
        delta_F <- as.matrix(F_curr - F_prev)

        # Prevent division by zero
        if (sum(abs(delta_F)) < 1e-8) {
            cat("Gradient diff is too small, stopping optimization.\n")
            break
        }

        # Ensure correct dimensions for Broyden update
        numerator <-
            (delta_theta - B_inv %*% delta_F) %*% t(delta_theta)
        # Convert scalar to numeric
        denominator <-
            as.numeric(t(delta_theta) %*% delta_F)

        # Updated inverse Jacobian approximation
        B_inv <-
            B_inv + numerator / denominator

        # Compute next theta using secant update
        theta_next <- theta_curr - alpha * B_inv %*% F_curr

        # Line search: Reduce step size if SSE increases
        while (sse(as.vector(theta_next), x, y) > sse(as.vector(theta_curr),
                                                      x, y)) {
            alpha <- alpha * step_reduction_factor
            theta_next <- theta_curr - alpha * B_inv %*% F_curr

            # Print progress
            # cat("Reducing step size to", alpha, "\n")
        }

        # Print intermediate results for debugging
        cat(
            sprintf(
                "Iteration %d: Theta = (%.4f, %.4f, %.4f), Alpha = %.4f\n",
                j,
                theta_next[1],
                theta_next[2],
                theta_next[3],
                alpha
            )
        )
```

```r
            # Check convergence
            if (sum(abs(theta_next - theta_curr)) < tol) {
                cat("Converged successfully.\n")
                break
            }

            # Update iterates
            theta_prev <- theta_curr
            theta_curr <- theta_next
        }

        return(as.vector(theta_curr))  # Convert back to numeric vector
    }

# Adjusted initial parameter guesses
theta0 <- c(2, 0.8,-1)   # Closer to true parameters
theta1 <- c(4, 1.2,-0.5)  # Slightly refined

# Run Improved Secant Method
estimated_theta <- secant_method_improved(theta0, theta1, x, y)
#> Iteration 1: Theta = (3.8521, 1.3054, 0.0057), Alpha = 0.0156
#> Iteration 2: Theta = (3.8521, 1.3054, 0.0057), Alpha = 0.0000
#> Converged successfully.

# Display estimated parameters
cat("Estimated parameters (A, B, C) using Secant Method:\n")
#> Estimated parameters (A, B, C) using Secant Method:
print(estimated_theta)
#> [1] 3.85213912 1.30538435 0.00566417

# Plot data and fitted curve
plot(
    x,
    y,
    main = "Secant Method: Data & Fitted Curve",
    pch = 19,
    cex = 0.5,
    xlab = "x",
    ylab = "y"
)
curve(
    nonlinear_model(estimated_theta, x),
    from = -5,
    to = 5,
    add = TRUE,
```

```
    col = "red",
    lwd = 2
)
legend(
    "topleft",
    legend = c("Data", "Fitted Curve"),
    pch = c(19, NA),
    lty = c(NA, 1),
    col = c("black", "red")
)
```

6.2.2.2 Grid Search

Grid Search (GS) is a brute-force optimization method that systematically explores a grid of possible parameter values to identify the combination that minimizes the residual sum of squares (RSS). Unlike gradient-based optimization, which moves iteratively towards a minimum, grid search evaluates all predefined parameter combinations, making it robust but computationally expensive.

Grid search is particularly useful when:

- **The function is highly nonlinear**, making gradient methods unreliable.
- **The parameter space is small**, allowing exhaustive search.
- **A global minimum is needed**, and prior knowledge about parameter ranges exists.

The goal of Grid Search is to find an optimal parameter set $\hat{\theta}$ that minimizes the Sum of Squared Errors (SSE):

$$\hat{\theta} = \arg\min_{\theta \in \Theta} SSE(\theta),$$

where:

$$SSE(\theta) = \sum_{i=1}^{n} (y_i - f(x_i; \theta))^2.$$

Grid search **discretizes** the search space Θ into a finite set of candidate values for each parameter:

$$\Theta = \theta_1 \times \theta_2 \times \cdots \times \theta_p.$$

The **accuracy** of the solution depends on the **grid resolution**—a finer grid leads to **better accuracy** but **higher computational cost**.

Grid Search Algorithm

1. Define a grid of possible values for each parameter.
2. Evaluate SSE for each combination of parameters.
3. Select the parameter set that minimizes SSE.

```r
# Define a numerically stable logistic function
safe_exp <- function(x) {
    return(ifelse(x > 700, Inf, exp(pmin(x, 700))))  # Prevent overflow
}

# Define the logistic growth model
nonlinear_model <- function(theta, x) {
    return(theta[1] / (1 + safe_exp(-theta[2] * (x - theta[3]))))
}

# Define the Sum of Squared Errors (SSE) function
sse <- function(theta, x, y) {
    predictions <- nonlinear_model(theta, x)
    return(sum((y - predictions)^2))
}

# Grid Search Optimization
grid_search_optimization <- function(x, y, grid_size = 10) {
```

```r
    # Define grid of parameter values
    A_values <- seq(2, 6, length.out = grid_size)
    B_values <- seq(0.5, 3, length.out = grid_size)
    C_values <- seq(-2, 2, length.out = grid_size)

    # Generate all combinations of parameters
    param_grid <- expand.grid(A = A_values, B = B_values, C = C_values)

    # Evaluate SSE for each parameter combination
    param_grid$SSE <- apply(param_grid, 1, function(theta) {
        sse(as.numeric(theta[1:3]), x, y)
    })

    # Select the best parameter set
    best_params <- param_grid[which.min(param_grid$SSE), 1:3]
    return(as.numeric(best_params))
}

# Generate synthetic data
set.seed(123)
x <- seq(-5, 5, length.out = 100)
true_theta <- c(4, 1.5, 0)  # True parameters (A, B, C)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.3)

# Run Grid Search
estimated_theta <- grid_search_optimization(x, y, grid_size = 20)

# Display results
cat("Estimated parameters (A, B, C) using Grid Search:\n")
#> Estimated parameters (A, B, C) using Grid Search:
print(estimated_theta)
#> [1] 4.1052632 1.4210526 0.1052632

# Plot data and fitted curve
plot(
    x, y,
    main = "Grid Search: Nonlinear Regression Optimization",
    pch = 19,
    cex = 0.5,
    xlab = "x",
    ylab = "y"
)
curve(
    nonlinear_model(estimated_theta, x),
    from = min(x),
```

```
    to = max(x),
    add = TRUE,
    col = "red",
    lwd = 2
)
legend(
    "topleft",
    legend = c("Data", "Fitted Curve"),
    pch = c(19, NA),
    lty = c(NA, 1),
    col = c("black", "red")
)
```

6.2.2.3 Nelder-Mead (Simplex)

The Nelder-Mead algorithm, also known as the Simplex method, is a derivative-free optimization algorithm that iteratively adjusts a geometric shape (simplex) to find the minimum of an objective function. It is particularly useful for non-linear regression when gradient-based methods fail due to non-differentiability or noisy function evaluations.

Nelder-Mead is particularly useful when:

- The function is non-differentiable or noisy.
- Gradient-based methods are unreliable.
- The parameter space is low-dimensional.

The goal of Nelder-Mead is to find an optimal parameter set $\hat{\theta}$ that minimizes the Sum of Squared Errors (SSE):

$$\hat{\theta} = \arg\min_{\theta} SSE(\theta),$$

where:

$$SSE(\theta) = \sum_{i=1}^{n} (y_i - f(x_i; \theta))^2.$$

1. Simplex Representation

The algorithm maintains a simplex, a geometric shape with $p+1$ vertices for a p-dimensional parameter space.

Each vertex represents a parameter vector:

$$S = \{\theta_1, \theta_2, \dots, \theta_{p+1}\}.$$

2. Simplex Operations

At each iteration, the algorithm **updates the simplex** based on the objective function values at each vertex:

1. **Reflection**: Reflect the worst point θ_h across the centroid.

$$\theta_r = \theta_c + \alpha(\theta_c - \theta_h)$$

2. **Expansion**: If reflection improves the objective, expand the step.

$$\theta_e = \theta_c + \gamma(\theta_r - \theta_c)$$

3. **Contraction**: If reflection worsens the objective, contract towards the centroid.

$$\theta_c = \theta_c + \rho(\theta_h - \theta_c)$$

4. **Shrink**: If contraction fails, shrink the simplex.

$$\theta_i = \theta_1 + \sigma(\theta_i - \theta_1), \quad \forall i > 1$$

where:

- $\alpha = 1$ (reflection coefficient),
- $\gamma = 2$ (expansion coefficient),
- $\rho = 0.5$ (contraction coefficient),
- $\sigma = 0.5$ (shrink coefficient).

The process continues until convergence.

Nelder-Mead Algorithm

1. Initialize a simplex with $p + 1$ vertices.
2. Evaluate SSE at each vertex.
3. Perform reflection, expansion, contraction, or shrink operations.
4. Repeat until convergence.

```r
# Load required library
library(stats)


# Define a numerically stable logistic function
safe_exp <- function(x) {
    return(ifelse(x > 700, Inf, exp(pmin(x, 700))))  # Prevent overflow
}

# Define the logistic growth model
nonlinear_model <- function(theta, x) {
    return(theta[1] / (1 + safe_exp(-theta[2] * (x - theta[3]))))
}

# Define the Sum of Squared Errors (SSE) function
sse <- function(theta, x, y) {
    predictions <- nonlinear_model(theta, x)
    return(sum((y - predictions) ^ 2))
}

# Nelder-Mead Optimization
nelder_mead_optimization <- function(x, y) {
    # Initial guess for parameters
    initial_guess <- c(2, 1, 0)

    # Run Nelder-Mead optimization
    result <- optim(
        par = initial_guess,
        fn = sse,
```

```r
        x = x,
        y = y,
        method = "Nelder-Mead",
        control = list(maxit = 500)
    )

    return(result$par)  # Return optimized parameters
}

# Generate synthetic data
set.seed(123)
x <- seq(-5, 5, length.out = 100)
true_theta <- c(4, 1.5, 0)  # True parameters (A, B, C)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.3)

# Run Nelder-Mead Optimization
estimated_theta <- nelder_mead_optimization(x, y)

# Display results
cat("Estimated parameters (A, B, C) using Nelder-Mead:\n")
#> Estimated parameters (A, B, C) using Nelder-Mead:
print(estimated_theta)
#> [1] 4.06873116 1.42759898 0.01119379

# Plot data and fitted curve
plot(
    x,
    y,
    main = "Nelder-Mead: Nonlinear Regression Optimization",
    pch = 19,
    cex = 0.5,
    xlab = "x",
    ylab = "y"
)
curve(
    nonlinear_model(estimated_theta, x),
    from = min(x),
    to = max(x),
    add = TRUE,
    col = "red",
    lwd = 2
)
legend(
    "topleft",
    legend = c("Data", "Fitted Curve"),
```

```
    pch = c(19, NA),
    lty = c(NA, 1),
    col = c("black", "red")
)
```

6.2.2.4 Powell's Method

Powell's Method is a derivative-free optimization algorithm that minimizes a function without using gradients. It works by iteratively refining a set of search directions to efficiently navigate the parameter space. Unlike Nelder-Mead (Simplex), which adapts a simplex, Powell's method builds an orthogonal basis of search directions.

Powell's method is particularly useful when:

- The function is non-differentiable or noisy.
- Gradient-based methods are unreliable.
- The parameter space is low-to-moderate dimensional.

The goal of Powell's Method is to find an optimal parameter set $\hat{\theta}$ that minimizes the Sum of Squared Errors (SSE):

$$\hat{\theta} = \arg\min_{\theta} SSE(\theta),$$

where:

$$SSE(\theta) = \sum_{i=1}^{n} (y_i - f(x_i; \theta))^2.$$

1. Search Directions

Powell's method **maintains a set of search directions** $\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_p$:

$$D = \{\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_p\}.$$

Initially, these directions are chosen as **unit basis vectors** (each optimizing a single parameter independently).

2. Line Minimization

For each direction $\mathbf{d}_j$, Powell's method **performs a 1D optimization**:

$$\theta' = \theta + \lambda \mathbf{d}_j,$$

where λ is chosen to minimize $SSE(\theta')$.

3. Updating Search Directions

After optimizing along all directions:

1. The **largest improvement** direction $\mathbf{d}_{\text{max}}$ is replaced with:

$$\mathbf{d}_{\text{new}} = \theta_{\text{final}} - \theta_{\text{initial}}.$$

2. The **new direction set is orthogonalized** using Gram-Schmidt.

This ensures **efficient exploration** of the parameter space.

4. Convergence Criteria

Powell's method **stops when function improvement is below a tolerance level**:

$$|SSE(\theta_{t+1}) - SSE(\theta_t)| < \epsilon.$$

Powell's Method Algorithm

1. Initialize search directions (standard basis vectors).
2. Perform 1D line searches along each direction.
3. Update the search directions based on the largest improvement.
4. Repeat until convergence.

```r
# Load required library
library(stats)

# Define a numerically stable logistic function
safe_exp <- function(x) {
    return(ifelse(x > 700, Inf, exp(pmin(x, 700))))  # Prevent overflow
}

# Define the logistic growth model
nonlinear_model <- function(theta, x) {
    return(theta[1] / (1 + safe_exp(-theta[2] * (x - theta[3]))))
}

# Define the Sum of Squared Errors (SSE) function
sse <- function(theta, x, y) {
    predictions <- nonlinear_model(theta, x)
    return(sum((y - predictions) ^ 2))
}

# Powell's Method Optimization
powell_optimization <- function(x, y) {
    # Initial guess for parameters
    initial_guess <- c(2, 1, 0)

    # Run Powell's optimization (via BFGS without gradient)
    result <- optim(
        par = initial_guess,
        fn = sse,
        x = x,
        y = y,
        method = "BFGS",
        control = list(maxit = 500),
        gr = NULL  # No gradient used
    )

    return(result$par)  # Return optimized parameters
}

# Generate synthetic data
set.seed(123)
x <- seq(-5, 5, length.out = 100)
true_theta <- c(4, 1.5, 0)  # True parameters (A, B, C)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.3)

# Run Powell's Method Optimization
```

```r
estimated_theta <- powell_optimization(x, y)

# Display results
cat("Estimated parameters (A, B, C) using Powell's Method:\n")
#> Estimated parameters (A, B, C) using Powell's Method:
print(estimated_theta)
#> [1] 4.06876538 1.42765687 0.01128753

# Plot data and fitted curve
plot(
    x,
    y,
    main = "Powell's Method: Nonlinear Regression Optimization",
    pch = 19,
    cex = 0.5,
    xlab = "x",
    ylab = "y"
)
curve(
    nonlinear_model(estimated_theta, x),
    from = min(x),
    to = max(x),
    add = TRUE,
    col = "red",
    lwd = 2
)
legend(
    "topleft",
    legend = c("Data", "Fitted Curve"),
    pch = c(19, NA),
    lty = c(NA, 1),
    col = c("black", "red")
)
```

6.2.2.5 Random Search

Random Search (RS) is a simple yet effective optimization algorithm that explores the search space by randomly sampling candidate solutions. Unlike grid search, which evaluates all predefined parameter combinations, random search selects a random subset, reducing computational cost.

Random search is particularly useful when:

- The search space is large, making grid search impractical.
- Gradient-based methods fail due to non-differentiability or noisy data.
- The optimal region is unknown, making global exploration essential.

The goal of Random Search is to find an optimal parameter set $\hat{\theta}$ that minimizes the Sum of Squared Errors (SSE):

$$\hat{\theta} = \arg\min_{\theta \in \Theta} SSE(\theta),$$

where:

$$SSE(\theta) = \sum_{i=1}^{n} (y_i - f(x_i; \theta))^2.$$

Unlike grid search, **random search does not evaluate all parameter combinations** but instead **randomly samples a subset**:

$$\Theta_{\text{sampled}} \subset \Theta.$$

The **accuracy** of the solution depends on the **number of random samples**—a higher number increases **the likelihood of finding a good solution**.

Random Search Algorithm

1. Define a random sampling range for each parameter.
2. Randomly sample N parameter sets.
3. Evaluate SSE for each sampled set.
4. Select the parameter set that minimizes SSE.

```r
# Load required library
library(stats)

# Define a numerically stable logistic function
safe_exp <- function(x) {
  return(ifelse(x > 700, Inf, exp(pmin(x, 700))))  # Prevent overflow
}

# Define the logistic growth model
nonlinear_model <- function(theta, x) {
  return(theta[1] / (1 + safe_exp(-theta[2] * (x - theta[3]))))
}

# Define the Sum of Squared Errors (SSE) function
sse <- function(theta, x, y) {
  predictions <- nonlinear_model(theta, x)
  return(sum((y - predictions) ^ 2))
}

# Random Search Optimization
random_search_optimization <- function(x, y, num_samples = 100) {
  # Define parameter ranges
  A_range <-
    runif(num_samples, 2, 6)   # Random values between 2 and 6
  B_range <-
    runif(num_samples, 0.5, 3) # Random values between 0.5 and 3
  C_range <-
    runif(num_samples,-2, 2)  # Random values between -2 and 2

  # Initialize best parameters
  best_theta <- NULL
```

```r
  best_sse <- Inf

  # Evaluate randomly sampled parameter sets
  for (i in 1:num_samples) {
    theta <- c(A_range[i], B_range[i], C_range[i])
    current_sse <- sse(theta, x, y)

    if (current_sse < best_sse) {
      best_sse <- current_sse
      best_theta <- theta
    }
  }

  return(best_theta)
}

# Generate synthetic data
set.seed(123)
x <- seq(-5, 5, length.out = 100)
true_theta <- c(4, 1.5, 0)  # True parameters (A, B, C)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.3)

# Run Random Search
estimated_theta <-
  random_search_optimization(x, y, num_samples = 500)

# Display results
cat("Estimated parameters (A, B, C) using Random Search:\n")
#> Estimated parameters (A, B, C) using Random Search:
print(estimated_theta)
#> [1] 4.0893431 1.4687456 0.1024474

# Plot data and fitted curve
plot(
  x,
  y,
  main = "Random Search: Nonlinear Regression Optimization",
  pch = 19,
  cex = 0.5,
  xlab = "x",
  ylab = "y"
)
curve(
  nonlinear_model(estimated_theta, x),
  from = min(x),
```

```
  to = max(x),
  add = TRUE,
  col = "red",
  lwd = 2
)
legend(
  "topleft",
  legend = c("Data", "Fitted Curve"),
  pch = c(19, NA),
  lty = c(NA, 1),
  col = c("black", "red")
)
```

6.2.2.6 Hooke-Jeeves Pattern Search

Hooke-Jeeves Pattern Search is a derivative-free optimization algorithm that searches for an optimal solution by exploring and adjusting parameter values iteratively. Unlike gradient-based methods, it does not require differentiability, making it effective for non-smooth and noisy functions.

Hooke-Jeeves is particularly useful when:

- The function is non-differentiable or noisy.
- Gradient-based methods are unreliable.
- The parameter space is low-to-moderate dimensional.

The goal of Hooke-Jeeves Pattern Search is to find an optimal parameter set $\hat{\theta}$ that minimizes the Sum of Squared Errors (SSE):

$$\hat{\theta} = \arg\min_{\theta} SSE(\theta),$$

where:

$$SSE(\theta) = \sum_{i=1}^{n} (y_i - f(x_i; \theta))^2.$$

1. Exploratory Moves

At each iteration, the algorithm **perturbs each parameter** to find a lower SSE:

$$\theta' = \theta \pm \delta.$$

If a new parameter θ' reduces SSE, it becomes the new base point.

2. Pattern Moves

After an improvement, the algorithm **accelerates movement** in the promising direction:

$$\theta_{\text{new}} = \theta_{\text{old}} + (\theta_{\text{old}} - \theta_{\text{prev}}).$$

This **speeds up convergence** towards an optimum.

3. Step Size Adaptation

If no improvement is found, the step size δ is reduced:

$$\delta_{\text{new}} = \beta \cdot \delta.$$

where $\beta < 1$ is a reduction factor.

4. Convergence Criteria

The algorithm **stops when step size is sufficiently small**:

$$\delta < \epsilon.$$

Hooke-Jeeves Algorithm

1. Initialize a starting point θ and step size δ.

2. Perform exploratory moves in each parameter direction.
3. If improvement is found, perform pattern moves.
4. Reduce step size if no improvement is found.
5. Repeat until convergence.

```r
# Load required library
library(stats)

# Define a numerically stable logistic function
safe_exp <- function(x) {
    return(ifelse(x > 700, Inf, exp(pmin(x, 700))))  # Prevent overflow
}

# Define the logistic growth model
nonlinear_model <- function(theta, x) {
    return(theta[1] / (1 + safe_exp(-theta[2] * (x - theta[3]))))
}

# Define the Sum of Squared Errors (SSE) function
sse <- function(theta, x, y) {
    predictions <- nonlinear_model(theta, x)
    return(sum((y - predictions) ^ 2))
}

# Hooke-Jeeves Pattern Search Optimization
hooke_jeeves_optimization <-
    function(x,
             y,
             step_size = 0.5,
             tol = 1e-6,
             max_iter = 500) {
        # Initial guess for parameters
        theta <- c(2, 1, 0)
        best_sse <- sse(theta, x, y)
        step <- step_size
        iter <- 0

        while (step > tol & iter < max_iter) {
            iter <- iter + 1
            improved <- FALSE
            new_theta <- theta

            # Exploratory move in each parameter direction
            for (i in 1:length(theta)) {
```

```
                for (delta in c(step,-step)) {
                    theta_test <- new_theta
                    theta_test[i] <- theta_test[i] + delta
                    sse_test <- sse(theta_test, x, y)

                    if (sse_test < best_sse) {
                        best_sse <- sse_test
                        new_theta <- theta_test
                        improved <- TRUE
                    }
                }
            }

            # Pattern move if improvement found
            if (improved) {
                theta <- 2 * new_theta - theta
                best_sse <- sse(theta, x, y)
            } else {
                step <- step / 2  # Reduce step size
            }
        }

        return(theta)
    }

# Generate synthetic data
set.seed(123)
x <- seq(-5, 5, length.out = 100)
true_theta <- c(4, 1.5, 0)  # True parameters (A, B, C)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.3)

# Run Hooke-Jeeves Optimization
estimated_theta <- hooke_jeeves_optimization(x, y)

# Display results
cat("Estimated parameters (A, B, C) using Hooke-Jeeves:\n")
#> Estimated parameters (A, B, C) using Hooke-Jeeves:
print(estimated_theta)
#> [1] 4 1 0

# Plot data and fitted curve
plot(
    x,
    y,
    main = "Hooke-Jeeves: Nonlinear Regression Optimization",
```

```
    pch = 19,
    cex = 0.5,
    xlab = "x",
    ylab = "y"
)
curve(
    nonlinear_model(estimated_theta, x),
    from = min(x),
    to = max(x),
    add = TRUE,
    col = "red",
    lwd = 2
)
legend(
    "topleft",
    legend = c("Data", "Fitted Curve"),
    pch = c(19, NA),
    lty = c(NA, 1),
    col = c("black", "red")
)
```

6.2.2.7 Bisection Method

The **Bisection Method** is a fundamental numerical technique primarily used for **root finding**, but it can also be adapted for **optimization problems** where the goal is to minimize or maximize a nonlinear function.

In nonlinear regression, optimization often involves **finding the parameter values that minimize the sum of squared errors (SSE)**:

$$\hat{\theta} = \arg\min_{\theta} SSE(\theta)$$

where:

$$SSE(\theta) = \sum_{i=1}^{n} \left(y_i - f(x_i; \theta)\right)^2.$$

Since the minimum of a function occurs where the derivative equals zero, we apply the **Bisection Method** to the **derivative of the SSE function**:

$$\frac{d}{d\theta} SSE(\theta) = 0.$$

This transforms the **optimization problem** into a **root-finding problem**.

The **Bisection Method** is based on the **Intermediate Value Theorem**, which states:

> If a continuous function $f(\theta)$ satisfies $f(\theta_a) \cdot f(\theta_b) < 0$,
> then there exists at least one root in the interval (θ_a, θ_b).

For **optimization**, we apply this principle to the **first derivative** of the objective function $Q(\theta)$:

$$Q'(\theta) = 0.$$

Step-by-Step Algorithm for Optimization

1. **Choose an interval** $[\theta_a, \theta_b]$ such that:

$$Q'(\theta_a) \cdot Q'(\theta_b) < 0.$$

2. **Compute the midpoint**:

$$\theta_m = \frac{\theta_a + \theta_b}{2}.$$

3. **Evaluate** $Q'(\theta_m)$:
   - If $Q'(\theta_m) = 0$, then θ_m is the optimal point.
   - If $Q'(\theta_a) \cdot Q'(\theta_m) < 0$, set $\theta_b = \theta_m$.
   - Otherwise, set $\theta_a = \theta_m$.

4. **Repeat** until convergence:

$$|\theta_b - \theta_a| < \epsilon.$$

Determining the Nature of the Critical Point

Since the Bisection Method finds a **stationary point**, we use the **second derivative test** to classify it:

- If $Q''(\theta) > 0$, the point is a **local minimum**.
- If $Q''(\theta) < 0$, the point is a **local maximum**.

For **nonlinear regression**, we expect $Q(\theta) = SSE(\theta)$, so the solution found by Bisection should be a **minimum**.

```r
# Load required library
library(stats)

# Define a numerically stable logistic function
safe_exp <- function(x) {
    return(ifelse(x > 700, Inf, exp(pmin(x, 700))))  # Prevent overflow
}

# Define the logistic growth model
nonlinear_model <- function(theta, x) {
    return(theta[1] / (1 + safe_exp(-theta[2] * (x - theta[3]))))
}

# Define the Sum of Squared Errors (SSE) function for optimization
sse <- function(theta, x, y) {
    predictions <- nonlinear_model(theta, x)
    return(sum((y - predictions)^2))
}

# Optimize all three parameters simultaneously
find_optimal_parameters <- function(x, y) {
    # Initial guess for parameters (based on data)
    initial_guess <- c(max(y), 1, median(x))

    # Bounds for parameters
    lower_bounds <- c(0.1, 0.01, min(x))  # Ensure positive scaling
    upper_bounds <- c(max(y) * 2, 10, max(x))

    # Run optim() with L-BFGS-B (bounded optimization)
    result <- optim(
        par = initial_guess,
        fn = sse,
```

```r
        x = x, y = y,
        method = "L-BFGS-B",
        lower = lower_bounds,
        upper = upper_bounds
    )

    return(result$par)  # Extract optimized parameters
}

# Generate synthetic data
set.seed(123)
x <- seq(-5, 5, length.out = 100)
true_theta <- c(4, 1.5, 0)  # True parameters (A, B, C)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.3)

# Find optimal parameters using optim()
estimated_theta <- find_optimal_parameters(x, y)

# Display results
cat("Estimated parameters (A, B, C) using optim():\n")
#> Estimated parameters (A, B, C) using optim():
print(estimated_theta)
#> [1] 4.06876536 1.42765688 0.01128756

# Plot data and fitted curve
plot(
    x, y,
    main = "Optim(): Nonlinear Regression Optimization",
    pch = 19,
    cex = 0.5,
    xlab = "x",
    ylab = "y"
)
curve(
    nonlinear_model(estimated_theta, x),
    from = min(x),
    to = max(x),
    add = TRUE,
    col = "red",
    lwd = 2
)
legend(
    "topleft",
    legend = c("Data", "Fitted Curve"),
    pch = c(19, NA),
```

```
    lty = c(NA, 1),
    col = c("black", "red")
)
```

6.2.3 Stochastic Heuristic

6.2.3.1 Differential Evolution Algorithm

The **Differential Evolution (DE) Algorithm** is a **stochastic, population-based optimization algorithm** that is widely used for solving complex **global optimization problems**. Unlike gradient-based methods such as Newton's method or the Secant method, DE does not require derivatives and is well-suited for optimizing **non-differentiable, nonlinear, and multimodal functions**.

Key Features of Differential Evolution

- **Population-based approach:** Maintains a population of candidate solutions instead of a single point.
- **Mutation and crossover:** Introduces variations to explore the search space.
- **Selection mechanism:** Retains the best candidates for the next generation.
- **Global optimization:** Avoids local minima by using **stochastic search strategies**.

Mathematical Formulation of Differential Evolution

Differential Evolution operates on a **population** of candidate solutions $\{\theta_i\}$, where each θ_i is a **vector of parameters**. The algorithm iteratively updates the population using three main operations:

1. Mutation

For each candidate solution θ_i, a **mutant vector** $\mathbf{v}_i^{(j)}$ is generated as:

$$\mathbf{v}_i^{(j)} = {}_{r_1}^{(j)} + F \cdot ({}_{r_2}^{(j)} - {}_{r_3}^{(j)})$$

where:

- ${}_{r_1}, {}_{r_2}, {}_{r_3}$ are **randomly selected distinct vectors** from the population.
- $F \in (0, 2)$ is the **mutation factor** controlling the step size.

2. Crossover

A trial vector $\mathbf{u}_i^{(j)}$ is generated by combining the mutant vector $\mathbf{v}_i^{(j)}$ with the original solution ${}_i^{(j)}$:

$$u_{i,k}^{(j)} = \begin{cases} v_{i,k}^{(j)} & \text{if } rand_k \leq C_r \text{ or } k = k_{\text{rand}}, \\ \theta_{i,k}^{(j)} & \text{otherwise.} \end{cases}$$

where:

- $C_r \in (0, 1)$ is the **crossover probability**.
- $rand_k$ is a **random value between 0 and 1**.
- k_{rand} ensures at least **one parameter is mutated**.

3. Selection

The new candidate solution is accepted only if it improves the objective function:

$${}_i^{(j+1)} = \begin{cases} \mathbf{u}_i^{(j)} & \text{if } Q(\mathbf{u}_i^{(j)}) < Q({}_i^{(j)}), \\ {}_i^{(j)} & \text{otherwise.} \end{cases}$$

where $Q(\theta)$ is the **objective function** (e.g., sum of squared errors in regression problems).

Algorithm: Differential Evolution for Nonlinear Optimization

1. Initialize a population of candidate solutions.
2. Evaluate the objective function for each candidate.
3. Mutate individuals using a difference strategy.
4. Apply crossover to create trial solutions.

5. Select individuals based on their fitness (objective function value).
6. Repeat until convergence (or a stopping criterion is met).

```r
# Load required library
library(DEoptim)

# Define a nonlinear function (logistic model)
nonlinear_model <- function(theta, x) {
    return(theta[1] / (1 + exp(-theta[2] * (x - theta[3]))))
}

# Define the Sum of Squared Errors (SSE) function
sse <- function(theta, x, y) {
    return(sum((y - nonlinear_model(theta, x))^2))
}

# Generate synthetic data
set.seed(123)
x <- seq(-5, 5, length.out = 100)
true_theta <- c(4, 1.5, 0)  # True parameters (A, B, C)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.3)

# Define the objective function for DEoptim
objective_function <- function(theta) {
    return(sse(theta, x, y))
}

# Define parameter bounds
lower_bounds <- c(0, 0, -5)
upper_bounds <- c(10, 5, 5)

# Run Differential Evolution Algorithm
de_result <-
    DEoptim(
        objective_function,
        lower_bounds,
        upper_bounds,
        DEoptim.control(
            NP = 50,
            itermax = 100,
            F = 0.8,
            CR = 0.9,
            trace = F
        )
```

```r
    )

# Extract optimized parameters
estimated_theta <- de_result$optim$bestmem

# Display estimated parameters
cat("Estimated parameters (A, B, C) using Differential Evolution:\n")
#> Estimated parameters (A, B, C) using Differential Evolution:
print(estimated_theta)
#>       par1       par2       par3
#> 4.06876562 1.42765614 0.01128768

# Plot data and fitted curve
plot(
    x,
    y,
    main = "Differential Evolution: Data & Fitted Curve",
    pch = 19,
    cex = 0.5,
    xlab = "x",
    ylab = "y"
)
curve(
    nonlinear_model(estimated_theta, x),
    from = -5,
    to = 5,
    add = TRUE,
    col = "red",
    lwd = 2
)
legend(
    "topleft",
    legend = c("Data", "Fitted Curve"),
    pch = c(19, NA),
    lty = c(NA, 1),
    col = c("black", "red")
)
```

6.2.3.2 Simulated Annealing

Simulated Annealing (SA) is a probabilistic global optimization algorithm inspired by annealing in metallurgy, where a material is heated and slowly cooled to remove defects. In optimization, SA gradually refines a solution by exploring the search space, allowing occasional jumps to escape local minima, before converging to an optimal solution.

Simulated Annealing is particularly useful when:

- The function is highly nonlinear and multimodal.
- Gradient-based methods struggle due to non-differentiability or poor initialization.
- A global minimum is needed, rather than a local one.

---

1. Energy Function (Objective Function)

The goal of SA is to minimize an objective function $Q(\theta)$. For nonlinear regression, this is the Sum of Squared Errors (SSE):

$$Q(\theta) = SSE(\theta) = \sum_{i=1}^{n} (y_i - f(x_i; \theta))^2.$$

2. Probability of Acceptance

At each step, SA randomly perturbs the parameters θ to create a **new candidate solution** θ' and evaluates the change in SSE:

$$\Delta Q = Q(\theta') - Q(\theta).$$

The **Metropolis Criterion** determines whether to accept the new solution:

$$P(\text{accept}) = \begin{cases} 1, & \text{if } \Delta Q < 0 \quad \text{(new solution improves fit)} \\ \exp\left(-\frac{\Delta Q}{T}\right), & \text{if } \Delta Q \geq 0 \quad \text{(accept with probability).} \end{cases}$$

where:

- T is the **temperature** that **gradually decreases** over iterations.
- **Worse solutions** are accepted **with small probability** to escape local minima.

3. Cooling Schedule

The **temperature** follows a cooling schedule:

$$T_k = \alpha T_{k-1},$$

where $\alpha \in (0, 1)$ is a decay factor that controls cooling speed.

Simulated Annealing Algorithm

1. Initialize parameters θ randomly.
2. Set an initial temperature T_0 and cooling rate α.
3. Repeat for max iterations:
   - Generate perturbed candidate θ'.
   - Compute $\Delta Q = Q(\theta') - Q(\theta)$.
   - Accept if $\Delta Q < 0$ or with probability $\exp(-\Delta Q/T)$.
   - Reduce temperature: $T \leftarrow \alpha T$.
4. Return the best solution found.

```r
# Load required library
library(stats)

# Define a numerically stable logistic function
safe_exp <- function(x) {
    return(ifelse(x > 700, Inf, exp(pmin(x, 700))))  # Prevent overflow
}
```

```r
# Define the logistic growth model
nonlinear_model <- function(theta, x) {
    return(theta[1] / (1 + safe_exp(-theta[2] * (x - theta[3]))))
}

# Define the Sum of Squared Errors (SSE) function
sse <- function(theta, x, y) {
    predictions <- nonlinear_model(theta, x)
    return(sum((y - predictions) ^ 2))
}

# Simulated Annealing Algorithm
simulated_annealing <-
    function(x,
             y,
             initial_theta,
             T_init = 1.0,
             alpha = 0.99,
             max_iter = 5000) {
        # Initialize parameters
        theta <- initial_theta
        best_theta <- theta
        best_sse <- sse(theta, x, y)
        T <- T_init  # Initial temperature

        for (iter in 1:max_iter) {
            # Generate new candidate solution (small random perturbation)
            theta_new <- theta + rnorm(length(theta), mean = 0, sd = T)

            # Compute new SSE
            sse_new <- sse(theta_new, x, y)

            # Compute change in SSE
            delta_Q <- sse_new - best_sse

            # Acceptance criteria
            if (delta_Q < 0 || runif(1) < exp(-delta_Q / T)) {
                theta <- theta_new
                best_sse <- sse_new
                best_theta <- theta_new
            }

            # Reduce temperature
            T <- alpha * T
```

```
            # Stopping condition (very low temperature)
            if (T < 1e-6)
                break
        }

        return(best_theta)
    }

# Generate synthetic data
set.seed(123)
x <- seq(-5, 5, length.out = 100)
true_theta <- c(4, 1.5, 0)  # True parameters (A, B, C)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.3)

# Initial guess
initial_theta <-
    c(runif(1, 1, 5), runif(1, 0.1, 3), runif(1,-2, 2))

# Run Simulated Annealing
estimated_theta <- simulated_annealing(x, y, initial_theta)

# Display results
cat("Estimated parameters (A, B, C) using Simulated Annealing:\n")
#> Estimated parameters (A, B, C) using Simulated Annealing:
print(estimated_theta)
#> [1] 4.07180419 1.41457906 0.01422147

# Plot data and fitted curve
plot(
    x,
    y,
    main = "Simulated Annealing: Nonlinear Regression Optimization",
    pch = 19,
    cex = 0.5,
    xlab = "x",
    ylab = "y"
)
curve(
    nonlinear_model(estimated_theta, x),
    from = min(x),
    to = max(x),
    add = TRUE,
    col = "red",
    lwd = 2
)
```

```
legend(
    "topleft",
    legend = c("Data", "Fitted Curve"),
    pch = c(19, NA),
    lty = c(NA, 1),
    col = c("black", "red")
)
```

6.2.3.3 Genetic Algorithm

Genetic Algorithms (GA) are a class of evolutionary algorithms inspired by the principles of natural selection and genetics. Unlike deterministic optimization techniques, GA evolves a population of candidate solutions over multiple generations, using genetic operators such as selection, crossover, and mutation.

GA is particularly useful when:

- The function is nonlinear, non-differentiable, or highly multimodal.
- Gradient-based methods fail due to rugged function landscapes.
- A global minimum is required, rather than a local one.

The goal of a Genetic Algorithm is to find an optimal solution $\hat{\theta}$ that minimizes an objective function:

$$\hat{\theta} = \arg\min_{\theta} SSE(\theta),$$

where:

$$SSE(\theta) = \sum_{i=1}^{n} (y_i - f(x_i; \theta))^2.$$

1. Population Representation

Each candidate solution (individual) is represented as a **chromosome**, which is simply a vector of parameters:

$$\theta = (\theta_1, \theta_2, \theta_3)$$

An entire **population** consists of multiple such solutions.

2. Selection

Each individual's fitness is evaluated using:

$$\text{Fitness}(\theta) = -SSE(\theta)$$

We use **Tournament Selection** or **Roulette Wheel Selection** to choose parents for reproduction.

3. Crossover (Recombination)

A new solution θ' is generated by **combining** two parents:

$$\theta' = \alpha\theta_{\text{parent1}} + (1 - \alpha)\theta_{\text{parent2}}, \quad \alpha \sim U(0, 1).$$

4. Mutation

Random small changes are introduced to **increase diversity**:

$$\theta'_i = \theta_i + \mathcal{N}(0, \sigma),$$

where $\mathcal{N}(0, \sigma)$ is a **small Gaussian perturbation**.

5. Evolutionary Cycle

The **algorithm iterates** through:

1. Selection
2. Crossover
3. Mutation
4. Survival of the fittest
5. Termination when convergence is reached.

```r
# Load required library
library(GA)

# Define a numerically stable logistic function
safe_exp <- function(x) {
    return(ifelse(x > 700, Inf, exp(pmin(x, 700))))  # Prevent overflow
}

# Define the logistic growth model
nonlinear_model <- function(theta, x) {
    return(theta[1] / (1 + safe_exp(-theta[2] * (x - theta[3]))))
}

# Define the Sum of Squared Errors (SSE) function for optimization
sse <- function(theta, x, y) {
    predictions <- nonlinear_model(theta, x)
    return(sum((y - predictions) ^ 2))
}

# Genetic Algorithm for Optimization
ga_optimization <- function(x, y) {
    # Define fitness function (negative SSE for maximization)
    fitness_function <- function(theta) {
        # GA maximizes fitness, so we use negative SSE
        return(-sse(theta, x, y))
    }

    # Set parameter bounds
    lower_bounds <- c(0.1, 0.01, min(x))  # Ensure positive scaling
    upper_bounds <- c(max(y) * 2, 10, max(x))

    # Run GA optimization
    ga_result <- ga(
        type = "real-valued",
        fitness = fitness_function,
        lower = lower_bounds,
        upper = upper_bounds,
        popSize = 50,
        # Population size
        maxiter = 200,
        # Max generations
        pmutation = 0.1,
        # Mutation probability
        monitor = FALSE
    )
```

```r
    return(ga_result@solution)  # Return optimized parameters
}

# Generate synthetic data
set.seed(123)
x <- seq(-5, 5, length.out = 100)
true_theta <- c(4, 1.5, 0)  # True parameters (A, B, C)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.3)

# Run Genetic Algorithm
estimated_theta <- ga_optimization(x, y)

# Display results
cat("Estimated parameters (A, B, C) using Genetic Algorithm:\n")
#> Estimated parameters (A, B, C) using Genetic Algorithm:
print(estimated_theta)
#>             x1       x2         x3
#> [1,] 4.066144 1.433886 0.00824126

# Plot data and fitted curve
plot(
    x,
    y,
    main = "Genetic Algorithm: Nonlinear Regression Optimization",
    pch = 19,
    cex = 0.5,
    xlab = "x",
    ylab = "y"
)
curve(
    nonlinear_model(estimated_theta, x),
    from = min(x),
    to = max(x),
    add = TRUE,
    col = "red",
    lwd = 2
)
legend(
    "topleft",
    legend = c("Data", "Fitted Curve"),
    pch = c(19, NA),
    lty = c(NA, 1),
    col = c("black", "red")
)
```

6.2.3.4 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population-based global optimization algorithm inspired by the social behavior of birds and fish schools. Instead of using genetic operators (like in Genetic Algorithms), PSO models particles (solutions) flying through the search space, adjusting their position based on their own experience and the experience of their neighbors.

PSO is particularly useful when:

- The function is nonlinear, noisy, or lacks smooth gradients.
- Gradient-based methods struggle due to non-differentiability.
- A global minimum is needed, rather than a local one.

The goal of PSO is to find an optimal solution $\hat{\theta}$ that minimizes an objective function:

$$\hat{\theta} = \arg\min_{\theta} SSE(\theta),$$

where:

$$SSE(\theta) = \sum_{i=1}^{n} (y_i - f(x_i; \theta))^2.$$

1. Particle Representation

Each **particle** represents a candidate solution:

$$\theta_i = (\theta_{i1}, \theta_{i2}, \theta_{i3})$$

where θ_{ij} is the jth parameter of particle i.

2. Particle Velocity and Position Updates

Each particle moves in the search space with **velocity** v_i, which is updated as:

$$v_i^{(t+1)} = \omega v_i^{(t)} + c_1 r_1 (p_i - \theta_i^{(t)}) + c_2 r_2 (g - \theta_i^{(t)})$$

where:

- ω is the **inertia weight** (controls exploration vs. exploitation),
- c_1, c_2 are **acceleration coefficients**,
- $r_1, r_2 \sim U(0, 1)$ are random numbers,
- p_i is the **particle's personal best position**,
- g is the **global best position**.

Then, the **position update** is:

$$\theta_i^{(t+1)} = \theta_i^{(t)} + v_i^{(t+1)}$$

This process continues until **convergence criteria** (like a max number of iterations or minimum error) is met.

Particle Swarm Optimization Algorithm

1. Initialize particles randomly within search bounds.
2. Set random initial velocities.
3. Evaluate SSE for each particle.
4. Update the personal and global best solutions.
5. Update velocities and positions using the update equations.
6. Repeat until convergence.

```r
# Load required library
library(pso)

# Define a numerically stable logistic function
safe_exp <- function(x) {
    return(ifelse(x > 700, Inf, exp(pmin(x, 700))))  # Prevent overflow
}
```

```r
# Define the logistic growth model
nonlinear_model <- function(theta, x) {
    return(theta[1] / (1 + safe_exp(-theta[2] * (x - theta[3]))))
}

# Define the Sum of Squared Errors (SSE) function for optimization
sse <- function(theta, x, y) {
    predictions <- nonlinear_model(theta, x)
    return(sum((y - predictions) ^ 2))
}

# Particle Swarm Optimization (PSO) for Nonlinear Regression
pso_optimization <- function(x, y) {
    # Define fitness function (minimize SSE)
    fitness_function <- function(theta) {
        return(sse(theta, x, y))
    }

    # Set parameter bounds
    lower_bounds <- c(0.1, 0.01, min(x))  # Ensure positive scaling
    upper_bounds <- c(max(y) * 2, 10, max(x))

    # Run PSO optimization
    pso_result <- psoptim(
        par = c(1, 1, 0),
        # Initial guess
        fn = fitness_function,
        lower = lower_bounds,
        upper = upper_bounds,
        control = list(maxit = 200, s = 50)  # 200 iterations, 50 particles
    )

    return(pso_result$par)  # Return optimized parameters
}

# Generate synthetic data
set.seed(123)
x <- seq(-5, 5, length.out = 100)
true_theta <- c(4, 1.5, 0)  # True parameters (A, B, C)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.3)

# Run Particle Swarm Optimization
estimated_theta <- pso_optimization(x, y)

# Display results
```

```r
cat("Estimated parameters (A, B, C) using Particle Swarm Optimization:\n")
#> Estimated parameters (A, B, C) using Particle Swarm Optimization:
print(estimated_theta)
#> [1] 4.06876562 1.42765613 0.01128767

# Plot data and fitted curve
plot(
    x,
    y,
    main = "Particle Swarm Optimization: Nonlinear Regression Optimization",
    pch = 19,
    cex = 0.5,
    xlab = "x",
    ylab = "y"
)
curve(
    nonlinear_model(estimated_theta, x),
    from = min(x),
    to = max(x),
    add = TRUE,
    col = "red",
    lwd = 2
)
legend(
    "topleft",
    legend = c("Data", "Fitted Curve"),
    pch = c(19, NA),
    lty = c(NA, 1),
    col = c("black", "red")
)
```

6.2.3.5 Evolutionary Strategies

Evolutionary Strategies (ES) are a class of evolutionary optimization algorithms that improve solutions by mutating and selecting individuals based on fitness. Unlike Genetic Algorithm, ES focuses on self-adaptive mutation rates and selection pressure rather than crossover. This makes ES particularly robust for continuous optimization problems like nonlinear regression.

ES is particularly useful when:

- The function is complex, noisy, or lacks smooth gradients.
- Gradient-based methods fail due to non-differentiability.
- An adaptive approach to exploration and exploitation is needed.

The goal of ES is to find an optimal solution $\hat{\theta}$ that minimizes an objective function:

$$\hat{\theta} = \arg\min_{\theta} SSE(\theta),$$

where:

$$SSE(\theta) = \sum_{i=1}^{n} (y_i - f(x_i; \theta))^2.$$

1. Population Representation

Each **individual** is a solution θ_i in the parameter space:

$$\theta_i = (\theta_{i1}, \theta_{i2}, \theta_{i3}).$$

The **population** consists of multiple individuals, each representing different candidate parameters.

2. Mutation

New candidate solutions are generated by **adding random noise**:

$$\theta_i' = \theta_i + \sigma \mathcal{N}(0, I),$$

where:

- σ is the **mutation step size**, which adapts over time.
- $\mathcal{N}(0, I)$ is a standard normal distribution.

3. Selection

ES employs (μ, λ)-selection:

- (μ, λ)-ES: Select the best μ solutions from λ offspring.
- $(\mu + \lambda)$-ES: Combine parents and offspring, selecting the top μ.

4. Step-Size Adaptation

Mutation strength σ **self-adapts** using the **1/5 success rule**:

$$\sigma_{t+1} = \begin{cases} \sigma_t / c, & \text{if success rate } > 1/5 \\ \sigma_t \cdot c, & \text{if success rate } < 1/5 \end{cases}$$

where $c > 1$ is a scaling factor.

Evolutionary Strategies Algorithm

1. Initialize a population of λ solutions with random parameters.
2. Set mutation step size σ.
3. Repeat for max iterations:
   - Generate λ offspring by mutating parent solutions.
   - Evaluate fitness (SSE) of each offspring.
   - Select the best μ solutions for the next generation.
   - Adapt mutation step size based on success rate.
4. Return the best solution found.

```r
# Load required library
library(DEoptim)

# Define a numerically stable logistic function
safe_exp <- function(x) {
    return(ifelse(x > 700, Inf, exp(pmin(x, 700))))  # Prevent overflow
}

# Define the logistic growth model
nonlinear_model <- function(theta, x) {
    return(theta[1] / (1 + safe_exp(-theta[2] * (x - theta[3]))))
}

# Define the Sum of Squared Errors (SSE) function for optimization
sse <- function(theta, x, y) {
    predictions <- nonlinear_model(theta, x)
    return(sum((y - predictions)^2))
}

# Evolutionary Strategies Optimization (Using Differential Evolution)
es_optimization <- function(x, y) {
    # Define fitness function (minimize SSE)
    fitness_function <- function(theta) {
        return(sse(theta, x, y))
    }

    # Set parameter bounds
    lower_bounds <- c(0.1, 0.01, min(x))  # Ensure positive scaling
    upper_bounds <- c(max(y) * 2, 10, max(x))

    # Run Differential Evolution (mimicking ES)
    es_result <- DEoptim(
        fn = fitness_function,
        lower = lower_bounds,
        upper = upper_bounds,
        # 50 individuals, 200 generations, suppress iteration output
        DEoptim.control(NP = 50, itermax = 200, trace = F)
    )

    return(es_result$optim$bestmem)  # Return optimized parameters
}

# Generate synthetic data
set.seed(123)
x <- seq(-5, 5, length.out = 100)
```

```r
true_theta <- c(4, 1.5, 0)  # True parameters (A, B, C)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.3)

# Run Evolutionary Strategies Optimization
estimated_theta <- es_optimization(x, y)

# Display results
cat("Estimated parameters (A, B, C) using Evolutionary Strategies:\n")
#> Estimated parameters (A, B, C) using Evolutionary Strategies:
print(estimated_theta)
#>       par1       par2       par3
#> 4.06876561 1.42765613 0.01128767

# Plot data and fitted curve
plot(
    x, y,
    main = "Evolutionary Strategies: Nonlinear Regression Optimization",
    pch = 19,
    cex = 0.5,
    xlab = "x",
    ylab = "y"
)
curve(
    nonlinear_model(estimated_theta, x),
    from = min(x),
    to = max(x),
    add = TRUE,
    col = "red",
    lwd = 2
)
legend(
    "topleft",
    legend = c("Data", "Fitted Curve"),
    pch = c(19, NA),
    lty = c(NA, 1),
    col = c("black", "red")
)
```

6.2.4 Linearization

6.2.4.1 Taylor Series Approximation

Taylor Series Approximation is a fundamental tool in nonlinear optimization, enabling local approximation of complex functions using polynomial expansions. It is widely used to linearize nonlinear models, facilitate derivative-based optimization, and derive Newton-type methods.

Taylor series approximation is particularly useful when:

- A nonlinear function is difficult to compute directly.
- Optimization requires local gradient and curvature information.
- A simpler, polynomial-based approximation improves computational efficiency.

Given a differentiable function $f(\theta)$, its **Taylor series expansion** around a point θ_0 is:

$$f(\theta) = f(\theta_0) + f'(\theta_0)(\theta - \theta_0) + \frac{1}{2}f''(\theta_0)(\theta - \theta_0)^2 + \mathcal{O}((\theta - \theta_0)^3).$$

For optimization, we often use:

1. **First-order approximation (Linear Approximation):**

$$f(\theta) \approx f(\theta_0) + f'(\theta_0)(\theta - \theta_0).$$

2. **Second-order approximation (Quadratic Approximation):**

$$f(\theta) \approx f(\theta_0) + f'(\theta_0)(\theta - \theta_0) + \frac{1}{2} f''(\theta_0)(\theta - \theta_0)^2.$$

For **gradient-based optimization**, we use the **Newton-Raphson update**:

$$\theta^{(k+1)} = \theta^{(k)} - [H_f(\theta^{(k)})]^{-1} \nabla f(\theta^{(k)}),$$

where:

- $\nabla f(\theta)$ is the **gradient** (first derivative),
- $H_f(\theta)$ is the **Hessian matrix** (second derivative).

For nonlinear regression, we approximate the **Sum of Squared Errors (SSE)**:

$$SSE(\theta) = \sum_{i=1}^{n} (y_i - f(x_i; \theta))^2.$$

1. First-Order Approximation (Gradient Descent)

The **gradient** of SSE w.r.t. parameters θ is:

$$\nabla SSE(\theta) = -2 \sum_{i=1}^{n} (y_i - f(x_i; \theta)) \nabla f(x_i; \theta).$$

Using **first-order Taylor approximation**, we update parameters via **gradient descent**:

$$\theta^{(k+1)} = \theta^{(k)} - \alpha \nabla SSE(\theta^{(k)}),$$

where α is the **learning rate**.

2. Second-Order Approximation (Newton's Method)

The **Hessian matrix** of SSE is:

$$H_{SSE}(\theta) = 2 \sum_{i=1}^{n} \nabla f(x_i; \theta) \nabla f(x_i; \theta)^T - 2 \sum_{i=1}^{n} (y_i - f(x_i; \theta)) H_f(x_i; \theta).$$

The **Newton-Raphson update** becomes:

$$\theta^{(k+1)} = \theta^{(k)} - H_{SSE}(\theta)^{-1} \nabla SSE(\theta).$$

```r
# Load required libraries
library(numDeriv)

# Define a numerically stable logistic function
safe_exp <- function(x) {
    return(ifelse(is.na(x) |
                      x > 700, Inf, exp(pmin(x, 700))))  # Prevent overflow
}

# Define the logistic growth model
nonlinear_model <- function(theta, x) {
    return(theta[1] / (1 + safe_exp(-theta[2] * (x - theta[3]))))
}

# Define the Sum of Squared Errors (SSE) function
sse <- function(theta, x, y) {
    predictions <- nonlinear_model(theta, x)
    return(sum((y - predictions) ^ 2, na.rm = TRUE))  # Avoid NA errors
}

# First-Order Approximation: Gradient Descent Optimization
gradient_descent <-
    function(x,
             y,
             alpha = 0.005,
             tol = 1e-6,
             max_iter = 5000) {
        theta <- c(2, 1, 0)  # Initial guess
        for (i in 1:max_iter) {
            grad_sse <-
                grad(function(t)
                    sse(t, x, y), theta)  # Compute gradient
            theta_new <-
                theta - alpha * grad_sse  # Update parameters

            if (sum(abs(theta_new - theta)) < tol)
                break  # Check convergence
            theta <- theta_new
        }
        return(theta)
    }

# Second-Order Approximation: Newton's Method with Regularization
newton_method <-
```

```r
    function(x,
             y,
             tol = 1e-6,
             max_iter = 100,
             lambda = 1e-4) {
        theta <- c(2, 1, 0)  # Initial guess
        for (i in 1:max_iter) {
            grad_sse <-
                grad(function(t)
                    sse(t, x, y), theta)  # Compute gradient
            hessian_sse <-
                hessian(function(t)
                    sse(t, x, y), theta)  # Compute Hessian

            # Regularize Hessian to avoid singularity
            hessian_reg <-
                hessian_sse + lambda * diag(length(theta))

            # Ensure Hessian is invertible
            if (is.na(det(hessian_reg)) ||
                det(hessian_reg) < 1e-10) {
                message("Singular Hessian found; increasing regularization.")
                lambda <- lambda * 10  # Increase regularization
                next
            }

            # Newton update
            theta_new <- theta - solve(hessian_reg) %*% grad_sse

            if (sum(abs(theta_new - theta)) < tol)
                break  # Check convergence
            theta <- theta_new
        }
        return(theta)
    }

# Generate synthetic data
set.seed(123)
x <- seq(-5, 5, length.out = 100)
true_theta <- c(4, 1.5, 0)  # True parameters (A, B, C)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.3)

# Run Gradient Descent
estimated_theta_gd <- gradient_descent(x, y)
```

```r
# Run Newton's Method with Regularization
estimated_theta_newton <- newton_method(x, y)

# Display results
cat("Estimated parameters (A, B, C) using Gradient Descent:\n")
#> Estimated parameters (A, B, C) using Gradient Descent:
print(estimated_theta_gd)
#> [1] 4.06876224 1.42766371 0.01128539

cat("Estimated parameters (A, B, C) using Newton's Method:\n")
#> Estimated parameters (A, B, C) using Newton's Method:
print(estimated_theta_newton)
#>            [,1]
#> [1,] 4.06876368
#> [2,] 1.42766047
#> [3,] 0.01128636

# Plot data and fitted curve
plot(
    x,
    y,
    main = "Taylor Series Approximation: Nonlinear Regression Optimization",
    pch = 19,
    cex = 0.5,
    xlab = "x",
    ylab = "y"
)
curve(
    nonlinear_model(estimated_theta_gd, x),
    from = min(x),
    to = max(x),
    add = TRUE,
    col = "blue",
    lwd = 2,
    lty = 2  # Dashed line to differentiate Gradient Descent
)
curve(
    nonlinear_model(estimated_theta_newton, x),
    from = min(x),
    to = max(x),
    add = TRUE,
    col = "red",
    lwd = 2
)
legend(
```

```
    "topleft",
    legend = c("Data", "Gradient Descent", "Newton's Method (Regularized)"),
    pch = c(19, NA, NA),
    lty = c(NA, 2, 1),
    col = c("black", "blue", "red")
)
```

6.2.4.2 Log-Linearization

Log-Linearization is a mathematical technique used to transform nonlinear models into linear models by taking the logarithm of both sides. This transformation simplifies parameter estimation and enables the use of linear regression techniques on originally nonlinear functions.

Log-linearization is particularly useful when:

- The model exhibits exponential, power-law, or logistic growth behavior.
- Linear regression methods are preferred over nonlinear optimization.
- A linearized version provides better interpretability and computational efficiency.

A **nonlinear model** can often be expressed in the form:

$$y = f(x; \theta).$$

Applying a **log transformation**, we obtain:

$$\log y = g(x; \theta),$$

where $g(x; \theta)$ is now **linear in parameters**. We then estimate θ using Ordinary Least Squares.

Example 1: Exponential Model

Consider an **exponential growth model**:

$$y = Ae^{Bx}.$$

Taking the natural logarithm:

$$\log y = \log A + Bx.$$

This is now **linear in** $\log y$, allowing estimation via **linear regression**.

Example 2: Power Law Model

For a **power law function**:

$$y = Ax^B.$$

Taking logs:

$$\log y = \log A + B \log x.$$

Again, this is **linearized**, making it solvable via **OLS regression**.

Log-Linearization Algorithm

1. Apply the logarithm transformation to the dependent variable.
2. Transform the equation into a linear form.
3. Use linear regression (OLS) to estimate parameters.
4. Convert parameters back to original scale if necessary.

```r
# Load required library
library(stats)

# Generate synthetic data for an exponential model
set.seed(123)
x <- seq(1, 10, length.out = 100)
```

```r
true_A <- 2
true_B <- 0.3
y <- true_A * exp(true_B * x) + rnorm(length(x), sd = 0.5)

# Apply logarithmic transformation
log_y <- log(y)

# Fit linear regression model
log_linear_model <- lm(log_y ~ x)

# Extract estimated parameters
estimated_B <- coef(log_linear_model)[2]  # Slope in log-space
estimated_A <-
    exp(coef(log_linear_model)[1])  # Intercept (back-transformed)

# Display results
cat("Estimated parameters (A, B) using Log-Linearization:\n")
#> Estimated parameters (A, B) using Log-Linearization:
print(c(estimated_A, estimated_B))
#> (Intercept)           x
#>   2.0012577   0.3001223

# Plot data and fitted curve
plot(
    x,
    y,
    main = "Log-Linearization: Nonlinear Regression Optimization",
    pch = 19,
    cex = 0.5,
    xlab = "x",
    ylab = "y"
)
curve(
    estimated_A * exp(estimated_B * x),
    from = min(x),
    to = max(x),
    add = TRUE,
    col = "red",
    lwd = 2
)
legend(
    "topleft",
    legend = c("Data", "Fitted Log-Linear Model"),
    pch = c(19, NA),
    lty = c(NA, 1),
```

```
    col = c("black", "red")
)
```

6.2.5 Hybrid

6.2.5.1 Adaptive Levenberg-Marquardt

The **Levenberg-Marquardt Algorithm (LMA)** is a powerful **nonlinear least squares optimization method** that adaptively combines:

- Gauss-Newton Algorithm for fast convergence near the solution.
- Steepest Descent (Gradient Descent) for stability when far from the solution.

The Adaptive Levenberg-Marquardt Algorithm further adjusts the damping parameter τ dynamically, making it more efficient in practice.

Given an objective function **Sum of Squared Errors (SSE)**:

$$SSE(\theta) = \sum_{i=1}^{n} (y_i - f(x_i; \theta))^2.$$

The **update rule for LMA** is:

$$\widehat{\theta}^{(j+1)} = \widehat{\theta}^{(j)} - \alpha_j [\mathbf{F}(\widehat{\theta}^{(j)})' \mathbf{F}(\widehat{\theta}^{(j)}) + \tau \mathbf{I}_{p \times p}] \frac{\partial \mathbf{Q}(\widehat{\theta}^{(j)})}{\partial \theta}.$$

where:

- τ is the **adaptive damping parameter**.
- $\mathbf{I}_{p\times p}$ is the **identity matrix**.
- $\mathbf{F}(\widehat{\theta}^{(j)})$ is the **Jacobian matrix** of partial derivatives.
- $\frac{\partial \mathbf{Q}(\widehat{\theta}^{(j)})}{\partial \theta}$ is the **gradient vector**.
- α_j is the **learning rate**.

The key **adaptation rule for** τ:

- **If the new step decreases SSE**, **reduce** τ:

$$\tau \leftarrow \tau/10.$$

- **Otherwise, increase** τ to ensure stability:

$$\tau \leftarrow 10\tau.$$

This adjustment **ensures a balance between stability and efficiency**.

Adaptive Levenberg-Marquardt Algorithm

1. Initialize parameters θ_0, damping factor τ.
2. Compute Jacobian $\mathbf{F}(\widehat{\theta}^{(j)})$.
3. Compute step direction using modified Gauss-Newton update.
4. Adjust τ dynamically:
   - Decrease τ if SSE improves.
   - Increase τ if SSE worsens.
5. Repeat until convergence.

```r
# Load required libraries
library(numDeriv)

# Define a numerically stable logistic function
safe_exp <- function(x) {
    return(ifelse(x > 700, Inf, exp(pmin(x, 700))))  # Prevent overflow
}

# Define the logistic growth model
nonlinear_model <- function(theta, x) {
    return(theta[1] / (1 + safe_exp(-theta[2] * (x - theta[3]))))
}
```

```r
# Define the Sum of Squared Errors (SSE) function
sse <- function(theta, x, y) {
    predictions <- nonlinear_model(theta, x)
    return(sum((y - predictions) ^ 2))
}

# Adaptive Levenberg-Marquardt Optimization
adaptive_lm_optimization <-
    function(x, y, tol = 1e-6, max_iter = 100) {
        theta <- c(2, 1, 0)  # Initial parameter guess
        tau <- 1e-3  # Initial damping parameter
        alpha <- 1  # Step size scaling
        iter <- 0

        while (iter < max_iter) {
            iter <- iter + 1

            # Compute Jacobian numerically
            J <- jacobian(function(t)
                nonlinear_model(t, x), theta)

            # Compute gradient of SSE
            residuals <- y - nonlinear_model(theta, x)
            grad_sse <- -2 * t(J) %*% residuals

            # Compute Hessian approximation
            H <- 2 * t(J) %*% J + tau * diag(length(theta))

            # Compute parameter update step
            delta_theta <- solve(H, grad_sse)

            # Trial step
            theta_new <- theta - alpha * delta_theta

            # Compute SSE for new parameters
            if (sse(theta_new, x, y) < sse(theta, x, y)) {
                # Accept step, decrease tau
                theta <- theta_new
                tau <- tau / 10
            } else {
                # Reject step, increase tau
                tau <- tau * 10
            }

            # Check convergence
```

```
            if (sum(abs(delta_theta)) < tol)
                break
        }

        return(theta)
    }

# Generate synthetic data
set.seed(123)
x <- seq(-5, 5, length.out = 100)
true_theta <- c(4, 1.5, 0)  # True parameters (A, B, C)
y <- nonlinear_model(true_theta, x) + rnorm(length(x), sd = 0.3)

# Run Adaptive Levenberg-Marquardt Optimization
estimated_theta <- adaptive_lm_optimization(x, y)

# Display results
cat("Estimated parameters (A, B, C) using Adaptive Levenberg-Marquardt:\n")
#> Estimated parameters (A, B, C) using Adaptive Levenberg-Marquardt:
print(estimated_theta)
#>            [,1]
#> [1,] 4.06876562
#> [2,] 1.42765612
#> [3,] 0.01128767

# Plot data and fitted curve
plot(
    x,
    y,
    main = "Adaptive Levenberg-Marquardt: Nonlinear Regression Optimization",
    pch = 19,
    cex = 0.5,
    xlab = "x",
    ylab = "y"
)
curve(
    nonlinear_model(estimated_theta, x),
    from = min(x),
    to = max(x),
    add = TRUE,
    col = "red",
    lwd = 2
)
legend(
    "topleft",
```

```
    legend = c("Data", "Fitted Curve"),
    pch = c(19, NA),
    lty = c(NA, 1),
    col = c("black", "red")
)
```

6.2.6 Comparison of Nonlinear Optimizers

```
# ALL-IN-ONE R SCRIPT COMPARING MULTIPLE NONLINEAR-REGRESSION OPTIMIZERS

library(minpack.lm) # nlsLM (Levenberg-Marquardt)
library(dfoptim)    # Powell (nmk), Hooke-Jeeves
library(nloptr)     # trust-region reflective
library(GA)         # genetic algorithm
library(DEoptim)    # differential evolution
library(GenSA)      # simulated annealing
library(pso)        # particle swarm
library(MASS)       # for ginv fallback
library(microbenchmark)
library(ggplot2)
library(dplyr)

# -- 1) DEFINE MODELS (SIMPLE VS COMPLEX) ---

# 3-parameter logistic
f_logistic <- function(theta, x) {
```

```
  A <- theta[1]
  B <- theta[2]
  C <- theta[3]
  A / (1 + exp(-B * (x - C)))
}
sse_logistic <-
  function(theta, x, y)
    sum((y - f_logistic(theta, x)) ^ 2)

# 4-parameter "extended" model
f_complex <- function(theta, x) {
  A <- theta[1]
  B <- theta[2]
  C <- theta[3]
  D <- theta[4]
  A / (1 + exp(-B * (x - C))) + D * exp(-0.5 * x)
}
sse_complex <-
  function(theta, x, y)
    sum((y - f_complex(theta, x)) ^ 2)

# Generate synthetic data
set.seed(123)
n <- 100
x_data <- seq(-5, 5, length.out = n)
# "simple" scenario
true_theta_simple <- c(4, 1.5, 0)
y_data_simple <-
  f_logistic(true_theta_simple, x_data) + rnorm(n, sd = 0.3)
# "complex" scenario
true_theta_complex <- c(4, 1.2, -1, 0.5)
y_data_complex <-
  f_complex(true_theta_complex, x_data) + rnorm(n, sd = 0.3)

# -- 2) OPTIMIZERS (EXCEPT BISECTION) ----
#
# All methods share signature:
#   FUN(par, x, y, sse_fn, model_fn, lower=NULL, upper=NULL, ...)
# Some do not strictly use lower/upper if unconstrained.

# 2.1 Gauss-Newton
gauss_newton_fit <- function(par,
                             x,
                             y,
                             sse_fn,
```

```
                              model_fn,
                              lower = NULL,
                              upper = NULL,
                              max_iter = 100,
                              tol = 1e-6)
{
  theta <- par
  for (iter in seq_len(max_iter)) {
    eps <- 1e-6
    nP  <- length(theta)
    Fmat <- matrix(0, nrow = length(x), ncol = nP)
    for (p in seq_len(nP)) {
      pert <- theta
      pert[p] <- pert[p] + eps
      Fmat[, p] <-
        (model_fn(pert, x) - model_fn(theta, x)) / eps
    }
    r <- y - model_fn(theta, x)
    delta <- tryCatch(
      solve(t(Fmat) %*% Fmat, t(Fmat) %*% r),
      error = function(e) {
        # fallback to pseudoinverse
        MASS::ginv(t(Fmat) %*% Fmat) %*% (t(Fmat) %*% r)
      }
    )
    theta_new <- theta + delta
    if (sum(abs(theta_new - theta)) < tol)
      break
    theta <- theta_new
  }
  theta
}

# 2.2 Modified Gauss-Newton (with step halving)
modified_gauss_newton_fit <- function(par,
                                      x,
                                      y,
                                      sse_fn,
                                      model_fn,
                                      lower = NULL,
                                      upper = NULL,
                                      max_iter = 100,
                                      tol = 1e-6)
{
  theta <- par
```

```r
  for (iter in seq_len(max_iter)) {
    eps <- 1e-6
    nP  <- length(theta)
    Fmat <- matrix(0, nrow = length(x), ncol = nP)
    for (p in seq_len(nP)) {
      pert <- theta
      pert[p] <- pert[p] + eps
      Fmat[, p] <-
        (model_fn(pert, x) - model_fn(theta, x)) / eps
    }
    r <- y - model_fn(theta, x)
    lhs <- t(Fmat) %*% Fmat
    rhs <- t(Fmat) %*% r
    delta <- tryCatch(
      solve(lhs, rhs),
      error = function(e)
        MASS::ginv(lhs) %*% rhs
    )
    sse_old <- sse_fn(theta, x, y)
    alpha <- 1
    for (k in 1:10) {
      new_sse <- sse_fn(theta + alpha * delta, x, y)
      if (new_sse < sse_old)
        break
      alpha <- alpha / 2
    }
    theta_new <- theta + alpha * delta
    if (sum(abs(theta_new - theta)) < tol)
      break
    theta <- theta_new
  }
  theta
}

# 2.3 Steepest Descent (Gradient Descent)
steepest_descent_fit <- function(par,
                                 x,
                                 y,
                                 sse_fn,
                                 model_fn,
                                 lower = NULL,
                                 upper = NULL,
                                 lr = 0.001,
                                 max_iter = 5000,
                                 tol = 1e-6)
```

```r
{
  theta <- par
  for (iter in seq_len(max_iter)) {
    eps <- 1e-6
    f0  <- sse_fn(theta, x, y)
    grad <- numeric(length(theta))
    for (p in seq_along(theta)) {
      pert <- theta
      pert[p] <- pert[p] + eps
      grad[p] <- (sse_fn(pert, x, y) - f0) / eps
    }
    theta_new <- theta - lr * grad
    if (sum(abs(theta_new - theta)) < tol)
      break
    theta <- theta_new
  }
  theta
}

# 2.4 Levenberg-Marquardt (nlsLM)
lm_fit <- function(par,
                   x,
                   y,
                   sse_fn,
                   model_fn,
                   lower = NULL,
                   upper = NULL,
                   form = c("simple", "complex"))
{
  form <- match.arg(form)
  if (form == "simple") {
    fit <- nlsLM(y ~ A / (1 + exp(-B * (x - C))),
                 start = list(A = par[1],
                              B = par[2],
                              C = par[3]))
  } else {
    fit <- nlsLM(y ~ A / (1 + exp(-B * (x - C))) + D * exp(-0.5 * x),
                 start = list(
                   A = par[1],
                   B = par[2],
                   C = par[3],
                   D = par[4]
                 ))
  }
  coef(fit)
```

```
}

# 2.5 Newton-Raphson (with numeric Hessian, fallback if singular)
newton_raphson_fit <- function(par,
                               x,
                               y,
                               sse_fn,
                               model_fn,
                               lower = NULL,
                               upper = NULL,
                               max_iter = 50,
                               tol = 1e-6)
{
  theta <- par
  for (i in seq_len(max_iter)) {
    eps <- 1e-6
    f0  <- sse_fn(theta, x, y)
    grad <- numeric(length(theta))
    for (p in seq_along(theta)) {
      pert <- theta
      pert[p] <- pert[p] + eps
      grad[p] <- (sse_fn(pert, x, y) - f0) / eps
    }
    Hess <- matrix(0, length(theta), length(theta))
    for (p in seq_along(theta)) {
      pert_p <- theta
      pert_p[p] <- pert_p[p] + eps
      f_p <- sse_fn(pert_p, x, y)
      for (q in seq_along(theta)) {
        pert_q <- pert_p
        pert_q[q] <- pert_q[q] + eps
        Hess[p, q] <- (sse_fn(pert_q, x, y) -
                         f_p - (f0 - sse_fn(theta, x, y))) / (eps ^
                                                                2)
      }
    }
    delta <- tryCatch(
      solve(Hess, grad),
      error = function(e)
        MASS::ginv(Hess) %*% grad
    )
    theta_new <- theta - delta
    if (sum(abs(theta_new - theta)) < tol)
      break
    theta <- theta_new
```

```
  }
  theta
}

# 2.6 Quasi-Newton (BFGS via optim)
quasi_newton_fit <- function(par,
                             x,
                             y,
                             sse_fn,
                             model_fn,
                             lower = NULL,
                             upper = NULL)
{
  fn <- function(pp)
    sse_fn(pp, x, y)
  res <- optim(par, fn, method = "BFGS")
  res$par
}

# 2.7 Trust-region reflective (nloptr)
trust_region_fit <- function(par,
                             x,
                             y,
                             sse_fn,
                             model_fn,
                             lower = NULL,
                             upper = NULL)
{
  # numeric gradient
  grad_numeric <- function(pp, eps = 1e-6) {
    g  <- numeric(length(pp))
    f0 <- sse_fn(pp, x, y)
    for (i in seq_along(pp)) {
      p2 <- pp
      p2[i] <- p2[i] + eps
      g[i] <- (sse_fn(p2, x, y) - f0) / eps
    }
    g
  }
  eval_f <- function(pp) {
    val <- sse_fn(pp, x, y)
    gr  <- grad_numeric(pp)
    list(objective = val, gradient = gr)
  }
  lb <- if (is.null(lower))
```

```
    rep(-Inf, length(par))
  else
    lower
  ub <- if (is.null(upper))
    rep(Inf, length(par))
  else
    upper
  res <- nloptr(
    x0 = par,
    eval_f = eval_f,
    lb = lb,
    ub = ub,
    opts = list(
      algorithm = "NLOPT_LD_TNEWTON",
      maxeval = 500,
      xtol_rel = 1e-6
    )
  )
  res$solution
}

# 2.8 Grid search
grid_search_fit <- function(par,
                            x,
                            y,
                            sse_fn,
                            model_fn,
                            lower = NULL,
                            upper = NULL,
                            grid_defs = NULL)
{
  if (is.null(grid_defs))
    stop("Must provide grid_defs for multi-parameter grid search.")
  g <- expand.grid(grid_defs)
  g$SSE <-
    apply(g, 1, function(rowp)
      sse_fn(as.numeric(rowp), x, y))
  best_idx <- which.min(g$SSE)
  as.numeric(g[best_idx, seq_along(grid_defs)])
}

# 2.9 Nelder-Mead
nelder_mead_fit <- function(par,
                            x,
                            y,
```

```
                              sse_fn,
                              model_fn,
                              lower = NULL,
                              upper = NULL)
{
  fn <- function(pp)
    sse_fn(pp, x, y)
  res <- optim(par, fn, method = "Nelder-Mead")
  res$par
}

# 2.10 Powell's method (dfoptim::nmk for unconstrained)
powell_fit <-
  function(par,
           x,
           y,
           sse_fn,
           model_fn,
           lower = NULL,
           upper = NULL) {
    fn <- function(pp)
      sse_fn(pp, x, y)
    dfoptim::nmk(par, fn)$par
  }

# 2.11 Hooke-Jeeves (dfoptim::hjkb)
hooke_jeeves_fit <-
  function(par,
           x,
           y,
           sse_fn,
           model_fn,
           lower = NULL,
           upper = NULL) {
    fn <- function(pp)
      sse_fn(pp, x, y)
    dfoptim::hjkb(par, fn)$par
  }

# 2.12 Random Search
random_search_fit <- function(par,
                              x,
                              y,
                              sse_fn,
                              model_fn,
```

```
                              lower,
                              upper,
                              max_iter = 2000,
                              ...)
{
  best_par <- NULL
  best_sse <- Inf
  dimp <- length(lower)
  for (i in seq_len(max_iter)) {
    candidate <- runif(dimp, min = lower, max = upper)
    val <- sse_fn(candidate, x, y)
    if (val < best_sse) {
      best_sse <- val
      best_par <- candidate
    }
  }
  best_par
}

# 2.13 Differential Evolution (DEoptim)
diff_evo_fit <- function(par,
                         x,
                         y,
                         sse_fn,
                         model_fn,
                         lower,
                         upper,
                         max_iter = 100,
                         ...)
{
  fn <- function(v)
    sse_fn(v, x, y)
  out <- DEoptim(fn,
                 lower = lower,
                 upper = upper,
                 DEoptim.control(NP = 50, itermax = max_iter, trace = F))
  out$optim$bestmem
}

# 2.14 Simulated Annealing (GenSA)
sim_anneal_fit <- function(par,
                           x,
                           y,
                           sse_fn,
                           model_fn,
```

```
                              lower = NULL,
                              upper = NULL,
                              ...)
{
  fn <- function(pp)
    sse_fn(pp, x, y)
  lb <- if (is.null(lower))
    rep(-Inf, length(par))
  else
    lower
  ub <- if (is.null(upper))
    rep(Inf, length(par))
  else
    upper
  # GenSA requires: GenSA(par, fn, lower, upper, control=list(...))
  out <-
    GenSA(
      par,
      fn,
      lower = lb,
      upper = ub,
      control = list(max.call = 10000)
    )
  out$par
}

# 2.15 Genetic Algorithm (GA)
genetic_fit <- function(par,
                        x,
                        y,
                        sse_fn,
                        model_fn,
                        lower,
                        upper,
                        max_iter = 100,
                        ...)
{
  fitness_fun <- function(pp)
    - sse_fn(pp, x, y)
  gares <- ga(
    type = "real-valued",
    fitness = fitness_fun,
    lower = lower,
    upper = upper,
    popSize = 50,
```

```
    maxiter = max_iter,
    run = 50
  )
  gares@solution[1,]
}

# 2.16 Particle Swarm (pso)
particle_swarm_fit <- function(par,
                               x,
                               y,
                               sse_fn,
                               model_fn,
                               lower,
                               upper,
                               max_iter = 100,
                               ...)
{
  fn <- function(pp)
    sse_fn(pp, x, y)
  res <- psoptim(
    par = (lower + upper) / 2,
    fn = fn,
    lower = lower,
    upper = upper,
    control = list(maxit = max_iter)
  )
  res$par
}


# -- 3) RUN METHOD WRAPPER ---
run_method <- function(method_name,
                       FUN,
                       par_init,
                       x,
                       y,
                       sse_fn,
                       model_fn,
                       lower = NULL,
                       upper = NULL,
                       ...)
{
  mb <- microbenchmark(result = {
    out <- FUN(par_init, x, y, sse_fn, model_fn, lower, upper, ...)
    out
```

```
  }, times = 1)
  final_par <-
    FUN(par_init, x, y, sse_fn, model_fn, lower, upper, ...)
  if (is.null(final_par)) {
    # e.g. placeholders that return NULL
    return(data.frame(
      Method = method_name,
      Parameters = "N/A",
      SSE = NA,
      Time_ms = NA
    ))
  }
  data.frame(
    Method     = method_name,
    Parameters = paste(round(final_par, 4), collapse = ", "),
    SSE        = round(sse_fn(final_par, x, y), 6),
    Time_ms    = median(mb$time) / 1e6
  )
}

# -- 4) MASTER FUNCTION TO COMPARE ALL METHODS (SIMPLE / COMPLEX) ---

compare_all_methods <- function(is_complex = FALSE) {
  if (!is_complex) {
    # SIMPLE (3-param logistic)
    x <- x_data
    y <- y_data_simple
    sse_fn   <- sse_logistic
    model_fn <- f_logistic
    init_par <- c(3, 1, 0.5)
    grid_defs <- list(
      A = seq(2, 6, length.out = 10),
      B = seq(0.5, 2, length.out = 10),
      C = seq(-1, 1, length.out = 10)
    )
    lower <- c(1, 0.1,-3)
    upper <- c(6, 3,  3)
    lm_form <- "simple"
  } else {
    # COMPLEX (4-param model)
    x <- x_data
    y <- y_data_complex
    sse_fn   <- sse_complex
    model_fn <- f_complex
    init_par <- c(3, 1,-0.5, 0.2)
```

```
  grid_defs <- list(
    A = seq(2, 6, length.out = 8),
    B = seq(0.5, 2, length.out = 8),
    C = seq(-2, 2, length.out = 8),
    D = seq(0, 2, length.out = 8)
  )
  lower <- c(1, 0.1,-3, 0)
  upper <- c(6, 3,  3, 2)
  lm_form <- "complex"
}

# RUN each method
out <- bind_rows(
  run_method(
    "Gauss-Newton",
    gauss_newton_fit,
    init_par,
    x,
    y,
    sse_fn,
    model_fn
  ),
  run_method(
    "Modified Gauss-Newton",
    modified_gauss_newton_fit,
    init_par,
    x,
    y,
    sse_fn,
    model_fn
  ),
  run_method(
    "Steepest Descent",
    steepest_descent_fit,
    init_par,
    x,
    y,
    sse_fn,
    model_fn
  ),
  run_method(
    "Levenberg-Marquardt (nlsLM)",
    lm_fit,
    init_par,
    x,
```

```
    y,
    sse_fn,
    model_fn,
    form = lm_form
  ),
  run_method(
    "Newton-Raphson",
    newton_raphson_fit,
    init_par,
    x,
    y,
    sse_fn,
    model_fn
  ),
  run_method(
    "Quasi-Newton (BFGS)",
    quasi_newton_fit,
    init_par,
    x,
    y,
    sse_fn,
    model_fn
  ),
  run_method(
    "Trust-region Reflective",
    trust_region_fit,
    init_par,
    x,
    y,
    sse_fn,
    model_fn,
    lower,
    upper
  ),
  run_method(
    "Grid Search",
    grid_search_fit,
    NULL,
    x,
    y,
    sse_fn,
    model_fn,
    grid_defs = grid_defs
  ),
  run_method(
```

```
    "Nelder-Mead",
    nelder_mead_fit,
    init_par,
    x,
    y,
    sse_fn,
    model_fn
  ),
  run_method("Powell's method",
             powell_fit,
             init_par,
             x,
             y,
             sse_fn,
             model_fn),
  run_method(
    "Hooke-Jeeves",
    hooke_jeeves_fit,
    init_par,
    x,
    y,
    sse_fn,
    model_fn
  ),
  run_method(
    "Random Search",
    random_search_fit,
    NULL,
    x,
    y,
    sse_fn,
    model_fn,
    lower,
    upper,
    max_iter = 1000
  ),
  run_method(
    "Differential Evolution",
    diff_evo_fit,
    NULL,
    x,
    y,
    sse_fn,
    model_fn,
    lower,
```

```
      upper,
      max_iter = 50
    ),
    run_method(
      "Simulated Annealing",
      sim_anneal_fit,
      init_par,
      x,
      y,
      sse_fn,
      model_fn,
      lower,
      upper
    ),
    run_method(
      "Genetic Algorithm",
      genetic_fit,
      NULL,
      x,
      y,
      sse_fn,
      model_fn,
      lower,
      upper,
      max_iter = 50
    ),
    run_method(
      "Particle Swarm",
      particle_swarm_fit,
      NULL,
      x,
      y,
      sse_fn,
      model_fn,
      lower,
      upper,
      max_iter = 50
    )
  )
  out
}

# -- 5) RUN & VISUALIZE ----

# Compare "simple" logistic (3 params)
```

```r
results_simple  <- compare_all_methods(is_complex = FALSE)
results_simple$Problem <- "Simple"

# Compare "complex" (4 params)
results_complex <- compare_all_methods(is_complex = TRUE)
results_complex$Problem <- "Complex"

# Combine
all_results <- rbind(results_simple, results_complex)
# print(all_results)
# DT::datatable(all_results)

# Example: SSE by method & problem
ggplot(all_results, aes(x = Method, y = log(SSE), fill = Problem)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal(base_size = 11) +
  labs(title = "Comparison of SSE by Method & Problem Complexity",
       x = "", y = "Log(Sum of Squared Errors)") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```r
# Example: Time (ms) by method & problem
ggplot(all_results, aes(x = Method, y = Time_ms, fill = Problem)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal(base_size = 11) +
  labs(title = "Comparison of Computation Time by Method & Problem Complexity",
       x = "", y = "Time (ms)") +
```

```
theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

6.3 Practical Considerations

For optimization algorithms to converge, they require good initial estimates of the parameters. The choice of starting values, constraints, and the complexity of the model all play a role in whether an optimization algorithm successfully finds a suitable solution.

6.3.1 Selecting Starting Values

Choosing good starting values can significantly impact the efficiency and success of optimization algorithms. Several approaches can be used:

- **Prior or theoretical information**: If prior knowledge about the parameters is available, it should be incorporated into the choice of initial values.
- **Grid search or graphical inspection of** $SSE(\theta)$: Evaluating the sum of squared errors (SSE) across a grid of possible values can help identify promising starting points.
- Ordinary Least Squares **estimates**: If a linear approximation of the model exists, using OLS to obtain initial estimates can be effective.
- **Model interpretation**: Understanding the structure and behavior of the model can provide intuition for reasonable starting values.
- **Expected Value Parameterization**: Reformulating the model based on expected values may improve the interpretability and numerical stability

of the estimation.

6.3.1.1 Grid Search for Optimal Starting Values

```
# Set seed for reproducibility
set.seed(123)

# Generate x as 100 integers using seq function
x <- seq(0, 100, 1)

# Generate coefficients for exponential function
a <- runif(1, 0, 20)  # Random coefficient a
b <- runif(1, 0.005, 0.075)  # Random coefficient b
c <- runif(101, 0, 5)  # Random noise

# Generate y as a * e^(b*x) + c
y <- a * exp(b * x) + c

# Print the generated parameters
cat("Generated coefficients:\n")
#> Generated coefficients:
cat("a =", a, "\n")
#> a = 5.75155
cat("b =", b, "\n")
#> b = 0.06018136

# Define our data frame
datf <- data.frame(x, y)

# Define our model function
mod <- function(a, b, x) {
  a * exp(b * x)
}
```

```
# Ensure all y values are positive (avoid log issues)
y_adj <-
  ifelse(y > 0, y, min(y[y > 0]) + 1e-3)  # Shift small values slightly

# Create adjusted dataframe
datf_adj <- data.frame(x, y_adj)

# Linearize by taking log(y)
lin_mod <- lm(log(y_adj) ~ x, data = datf_adj)

# Extract starting values
astrt <-
```

```r
  exp(coef(lin_mod)[1])  # Convert intercept back from log scale
bstrt <- coef(lin_mod)[2]  # Slope remains the same
cat("Starting values for non-linear fit:\n")
print(c(astrt, bstrt))

# Fit nonlinear model with these starting values
nlin_mod <- nls(y ~ mod(a, b, x),
                start = list(a = astrt, b = bstrt),
                data = datf)

# Model summary
summary(nlin_mod)

# Plot original data
plot(
  x,
  y,
  main = "Exponential Growth Fit",
  col = "blue",
  pch = 16,
  xlab = "x",
  ylab = "y"
)

# Add fitted curve in red
lines(x, predict(nlin_mod), col = "red", lwd = 2)

# Add legend
legend(
  "topleft",
  legend = c("Original Data", "Fitted Model"),
  col = c("blue", "red"),
  pch = c(16, NA),
  lwd = c(NA, 2)
)
```

```r
# Define grid of possible parameter values
aseq <- seq(10, 18, 0.2)
bseq <- seq(0.001, 0.075, 0.001)

na <- length(aseq)
nb <- length(bseq)
SSout <- matrix(0, na * nb, 3)  # Matrix to store SSE values
cnt <- 0
```

```r
# Evaluate SSE across grid
for (k in 1:na) {
  for (j in 1:nb) {
    cnt <- cnt + 1
    ypred <-
      # Evaluate model at these parameter values
      mod(aseq[k], bseq[j], x)

    # Compute SSE
    ss <- sum((y - ypred) ^ 2)
    SSout[cnt, 1] <- aseq[k]
    SSout[cnt, 2] <- bseq[j]
    SSout[cnt, 3] <- ss
  }
}

# Identify optimal starting values
mn_indx <- which.min(SSout[, 3])
astrt <- SSout[mn_indx, 1]
bstrt <- SSout[mn_indx, 2]

# Fit nonlinear model using optimal starting values
nlin_modG <-
  nls(y ~ mod(a, b, x), start = list(a = astrt, b = bstrt))

# Display model results
summary(nlin_modG)
#>
#> Formula: y ~ mod(a, b, x)
#>
#> Parameters:
#>    Estimate Std. Error t value Pr(>|t|)
#> a 5.889e+00  1.986e-02   296.6   <2e-16 ***
#> b 5.995e-02  3.644e-05  1645.0   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 2.135 on 99 degrees of freedom
#>
#> Number of iterations to convergence: 4
#> Achieved convergence tolerance: 7.204e-06
```

Note: The `nls_multstart` package can perform a grid search more efficiently without requiring manual looping.

Visualizing Prediction Intervals

Once the model is fitted, it is useful to visualize prediction intervals to assess model uncertainty.

```
# Load necessary package
library(nlstools)

# Plot fitted model with confidence and prediction intervals
plotFit(
  nlin_modG,
  interval = "both",
  pch = 19,
  shade = TRUE,
  col.conf = "skyblue4",
  col.pred = "lightskyblue2",
  data = datf
)
```

6.3.1.2 Using Programmed Starting Values in `nls`

Many nonlinear models have well-established functional forms, allowing for programmed starting values in the `nls` function. For example, models such as **logistic growth** and **asymptotic regression** have built-in self-starting functions.

To explore available self-starting models in R, use:

```
apropos("^SS")
```

This command lists functions with names starting with `SS`, which typically de-

note self-starting functions for nonlinear regression.

6.3.1.3 Custom Self-Starting Functions

If your model does not match any built-in `nls` functions, you can define your own **self-starting function**. Self-starting functions in `R` automate the process of estimating initial values, which helps in fitting nonlinear models more efficiently.

If needed, a self-starting function should:

1. Define the nonlinear equation.
2. Implement a method for computing starting values.
3. Return the function structure in an appropriate format.

6.3.2 Handling Constrained Parameters

In some cases, parameters must satisfy constraints (e.g., $\theta_i > a$ or $a < \theta_i < b$). The following strategies help address constrained parameter estimation:

1. **Fit the model without constraints first**: If the unconstrained parameter estimates satisfy the desired constraints, no further action is needed.
2. **Re-parameterization**: If the estimated parameters violate constraints, consider re-parameterizing the model to naturally enforce the required bounds.

6.3.3 Failure to Converge

Several factors can cause an algorithm to fail to converge:

- **A "flat" SSE function**: If the sum of squared errors $SSE(\theta)$ is relatively constant in the neighborhood of the minimum, the algorithm may struggle to locate an optimal solution.
- **Poor starting values**: Trying different or better initial values can help.
- **Overly complex models**: If the model is too complex relative to the data, consider simplifying it.

6.3.4 Convergence to a Local Minimum

- **Linear least squares models** have a well-defined, unique minimum because the SSE function is quadratic:

$$SSE(\theta) = (Y - X\beta)'(Y - X\beta)$$

- **Nonlinear least squares models** may have multiple local minima.
- **Testing different starting values** can help identify a global minimum.
- **Graphing** $SSE(\theta)$ as a function of individual parameters (if feasible) can provide insights.

- **Alternative optimization algorithms** such as Genetic Algorithm or particle swarm optimization may be useful in non-convex problems.

6.3.5 Model Adequacy and Estimation Considerations

Assessing the adequacy of a **nonlinear model** involves checking its **nonlinearity**, **goodness of fit**, and **residual behavior**. Unlike linear models, nonlinear models do not always have a direct equivalent of R^2, and issues such as collinearity, leverage, and residual heteroscedasticity must be carefully evaluated.

6.3.5.1 Components of Nonlinearity

Bates and Watts (1980) defines two key aspects of **nonlinearity** in statistical modeling:

1. **Intrinsic Nonlinearity**

- Measures the **bending and twisting** in the function $f(\theta)$.
- Assumes that the function is relatively **flat (planar)** in the neighborhood of $\hat{\theta}$.
- If severe, the **distribution of residuals** will be **distorted**.
- Leads to:
  - **Slow convergence** of optimization algorithms.
  - **Difficulties in identifying** parameter estimates.
- Solution approaches:
  - Higher-order **Taylor expansions** for estimation.
  - **Bayesian methods** for parameter estimation.

```r
# Check intrinsic curvature
modD <- deriv3(~ a * exp(b * x), c("a", "b"), function(a, b, x) NULL)

nlin_modD <- nls(y ~ modD(a, b, x),
                 start = list(a = astrt, b = bstrt),
                 data = datf)

rms.curv(nlin_modD)  # Function from the MASS package to assess curvature
#> Parameter effects: c^theta x sqrt(F) = 0.0564
#>         Intrinsic: c^iota  x sqrt(F) = 9e-04
```

2. **Parameter-Effects Nonlinearity**

- Measures **how the curvature** (nonlinearity) depends on the parameterization.
- Strong parameter effects nonlinearity can cause **problems with inference on** $\hat{\theta}$.
- Can be assessed using:

  - `rms.curv` function from `MASS`.
  - Bootstrap-based inference.

- **Solution:** Try **reparameterization** to stabilize the function.

6.3.5.2 Goodness of Fit in Nonlinear Models

In **linear regression**, we use the standard **coefficient of determination** (R^2):

$$R^2 = \frac{SSR}{SSTO} = 1 - \frac{SSE}{SSTO}$$

where:

- SSR = Regression Sum of Squares
- SSE = Error Sum of Squares
- $SSTO$ = Total Sum of Squares

However, in **nonlinear models**, the error and model sum of squares do not necessarily add up to the total corrected sum of squares:

$$SSR + SSE \neq SST$$

Thus, R^2 is not directly valid in the nonlinear case. Instead, we use a pseudo-R^2:

$$R^2_{pseudo} = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y})^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

- Unlike true R^2, this cannot be interpreted as the proportion of variability explained by the model.
- Should be used only for relative model comparison (e.g., comparing different nonlinear models).

6.3.5.3 Residual Analysis in Nonlinear Models

Residual plots help assess model adequacy, particularly when **intrinsic curvature is small**.

In nonlinear models, the **studentized residuals** are:

$$r_i = \frac{e_i}{s\sqrt{1 - \hat{c}_i}}$$

where:

- e_i = residual for observation i
- $\hat{c}_i$ = ith diagonal element of the **tangent-plane hat matrix**:

$$\hat{\mathbf{H}} = \mathbf{F}(\hat{\ })[\mathbf{F}(\hat{\ })'\mathbf{F}(\hat{\ })]^{-1}\mathbf{F}(\hat{\ })'$$

```
# Residual diagnostics for nonlinear models
library(nlstools)
resid_nls <- nlsResiduals(nlin_modD)

# Generate residual plots
plot(resid_nls)
```

6.3.5.4 Potential Issues in Nonlinear Regression Models

6.3.5.4.1 Collinearity

- Measures how correlated the model's predictors are.
- In nonlinear models, collinearity is assessed using the condition number of:

$$[\mathbf{F}(\hat{\ })'\mathbf{F}(\hat{\ })]^{-1}$$

- If condition number > 30, collinearity is a concern.
- Solution: Consider reparameterization (Magel and Hertsgaard, 1987).

6.3.5.4.2 Leverage

- Similar to leverage in Ordinary Least Squares.
- In nonlinear models, leverage is assessed using the **tangent-plane hat matrix**:

$$\hat{\mathbf{H}} = \mathbf{F}(\hat{\ })[\mathbf{F}(\hat{\ })'\mathbf{F}(\hat{\ })]^{-1}\mathbf{F}(\hat{\ })'$$

- **Solution:** Identify influential points and assess their impact on parameter estimates (St Laurent and Cook, 1992).

6.3.5.4.3 Heterogeneous Errors

- Non-constant variance across observations.
- **Solution:** Use **Weighted Nonlinear Least Squares (WNLS)**.

6.3.5.4.4 Correlated Errors

- Residuals may be autocorrelated.
- **Solution approaches:**
  - **Generalized Nonlinear Least Squares (GNLS)**
  - **Nonlinear Mixed Models (NLMEM)**
  - **Bayesian Methods**

Issue	Description	Solution
Intrinsic Nonlinearity	Function curvature independent of parameterization	Bayesian estimation, Taylor expansion
Parameter-Effects Nonlinearity	Curvature influenced by parameterization	Reparameterization, bootstrap
Collinearity	High correlation among predictors	Reparameterization, condition number check
Leverage	Influential points affecting model fit	Assess tangent-plane hat matrix
Heterogeneous Errors	Unequal variance in residuals	Weighted Nonlinear Least Squares
Correlated Errors	Autocorrelated residuals	GNLS, Nonlinear Mixed Models, Bayesian Methods

6.4 Application

6.4.1 Nonlinear Estimation Using Gauss-Newton Algorithm

This section demonstrates nonlinear parameter estimation using the Gauss-Newton algorithm and compares results with `nls()`. The model is given by:

$$y_i = \frac{\theta_0 + \theta_1 x_i}{1 + \theta_2 \exp(0.4 x_i)} + \epsilon_i$$

where

- $i = 1, \ldots, n$
- θ_0, θ_1, θ_2 are the unknown parameters.
- ϵ_i represents errors.

1. Loading and Visualizing the Data

```
library(dplyr)
library(ggplot2)

# Load the dataset
my_data <- read.delim("images/S21hw1pr4.txt", header = FALSE, sep = "") %>%
  dplyr::rename(x = V1, y = V2)

# Plot data
ggplot(my_data, aes(x = x, y = y)) +
  geom_point(color = "blue") +
  labs(title = "Observed Data", x = "X", y = "Y") +
  theme_minimal()
```

2. Deriving Starting Values for Parameters

Since nonlinear optimization is sensitive to starting values, we estimate reasonable initial values based on model interpretation.

Finding the Maximum Y Value

```
max(my_data$y)
#> [1] 2.6722
my_data$x[which.max(my_data$y)]
#> [1] 0.0094
```

- When $y = 2.6722$, the corresponding $x = 0.0094$.
- From the model equation: $\theta_0 + 0.0094\theta_1 = 2.6722$

Estimating θ_2 from the Median y Value

- The equation simplifies to: $1 + \theta_2 \exp(0.4x) = 2$

```
# find mean y
mean(my_data$y)
#> [1] -0.0747864

# find y closest to its mean
my_data$y[which.min(abs(my_data$y - (mean(my_data$y))))]
#> [1] -0.0773


# find x closest to the mean y
```

```r
my_data$x[which.min(abs(my_data$y - (mean(my_data$y))))]
#> [1] 11.0648
```

- This yields the equation: $83.58967\theta_2 = 1$

Finding the Value of θ_0 and θ_1

```r
# find value of x closet to 1
my_data$x[which.min(abs(my_data$x - 1))]
#> [1] 0.9895

# find index of x closest to 1
match(my_data$x[which.min(abs(my_data$x - 1))], my_data$x)
#> [1] 14

# find y value
my_data$y[match(my_data$x[which.min(abs(my_data$x - 1))], my_data$x)]
#> [1] 1.4577
```

- This provides another equation: $\theta_0 + \theta_1 \times 0.9895 - 2.164479\theta_2 = 1.457$

Solving for $\theta_0, \theta_1, \theta_2$

```r
library(matlib)

# Define coefficient matrix
A = matrix(
  c(0, 0.0094, 0, 0, 0, 83.58967, 1, 0.9895, -2.164479),
  nrow = 3,
  ncol = 3,
  byrow = T
)

# Define constant vector
b <- c(2.6722, 1, 1.457)

# Display system of equations
showEqn(A, b)
#> 0*x1 + 0.0094*x2        + 0*x3  =  2.6722
#> 0*x1      + 0*x2 + 83.58967*x3  =       1
#> 1*x1 + 0.9895*x2 - 2.164479*x3  =   1.457

# Solve for parameters
theta_start <- Solve(A, b, fractions = FALSE)
#> x1      =  -279.80879739
#>   x2    =   284.27659574
#>     x3  =      0.0119632
```

```
theta_start
#> [1] "x1      =  -279.80879739" "  x2    =   284.27659574"
#> [3] "    x3  =      0.0119632"
```

3. Implementing the Gauss-Newton Algorithm

Using these estimates, we manually implement the **Gauss-Newton optimization**.

Defining the Model and Its Derivatives

```
# Starting values
theta_0_strt <- as.numeric(gsub(".*=\\s*", "", theta_start[1]))
theta_1_strt <- as.numeric(gsub(".*=\\s*", "", theta_start[2]))
theta_2_strt <- as.numeric(gsub(".*=\\s*", "", theta_start[3]))

# Model function
mod_4 <- function(theta_0, theta_1, theta_2, x) {
  (theta_0 + theta_1 * x) / (1 + theta_2 * exp(0.4 * x))
}

# Define function expression
f_4 = expression((theta_0 + theta_1 * x) / (1 + theta_2 * exp(0.4 * x)))

# First derivatives
df_4.d_theta_0 <- D(f_4, 'theta_0')
df_4.d_theta_1 <- D(f_4, 'theta_1')
df_4.d_theta_2 <- D(f_4, 'theta_2')
```

Iterative Gauss-Newton Optimization

```
# Initialize
theta_vec <- matrix(c(theta_0_strt, theta_1_strt, theta_2_strt))
delta <- matrix(NA, nrow = 3, ncol = 1)
i <- 1

# Evaluate function at initial estimates
f_theta <- as.matrix(eval(f_4, list(
  x = my_data$x,
  theta_0 = theta_vec[1, 1],
  theta_1 = theta_vec[2, 1],
  theta_2 = theta_vec[3, 1]
)))

repeat {
  # Compute Jacobian matrix
  F_theta_0 <- as.matrix(cbind(
    eval(df_4.d_theta_0, list(
```

```
    x = my_data$x,
    theta_0 = theta_vec[1, i],
    theta_1 = theta_vec[2, i],
    theta_2 = theta_vec[3, i]
  )),
  eval(df_4.d_theta_1, list(
    x = my_data$x,
    theta_0 = theta_vec[1, i],
    theta_1 = theta_vec[2, i],
    theta_2 = theta_vec[3, i]
  )),
  eval(df_4.d_theta_2, list(
    x = my_data$x,
    theta_0 = theta_vec[1, i],
    theta_1 = theta_vec[2, i],
    theta_2 = theta_vec[3, i]
  ))
))

# Compute parameter updates
delta[, i] = (solve(t(F_theta_0)%*%F_theta_0))%*%t(F_theta_0)%*%(my_data$y-f_theta[,i])


# Update parameter estimates
theta_vec <- cbind(theta_vec, theta_vec[, i] + delta[, i])
theta_vec[, i + 1] = theta_vec[, i] + delta[, i]

# Increment iteration counter
i <- i + 1

# Compute new function values
f_theta <- cbind(f_theta, as.matrix(eval(f_4, list(
  x = my_data$x,
  theta_0 = theta_vec[1, i],
  theta_1 = theta_vec[2, i],
  theta_2 = theta_vec[3, i]
))))

delta = cbind(delta, matrix(NA, nrow = 3, ncol = 1))

# Convergence criteria based on SSE
if (abs(sum((my_data$y - f_theta[, i])^2) - sum((my_data$y - f_theta[, i - 1])^2)) /
    sum((my_data$y - f_theta[, i - 1])^2) < 0.001) {
  break
}
```

```
}

# Final parameter estimates
theta_vec[, ncol(theta_vec)]
#> [1]  3.6335135 -1.3055166  0.5043502
```

4. Checking Convergence and Variance

```
# Final objective function value (SSE)
sum((my_data$y - f_theta[, i])^2)
#> [1] 19.80165

sigma2 <- 1 / (nrow(my_data) - 3) *
  (t(my_data$y - f_theta[, ncol(f_theta)]) %*%
   (my_data$y - f_theta[, ncol(f_theta)]))  # p = 3

# Asymptotic variance-covariance matrix
as.numeric(sigma2)*as.matrix(solve(crossprod(F_theta_0)))
#>             [,1]        [,2]        [,3]
#> [1,]  0.11552571 -0.04817428  0.02685848
#> [2,] -0.04817428  0.02100861 -0.01158212
#> [3,]  0.02685848 -0.01158212  0.00703916
```

5. Validating with `nls()`

```
nlin_4 <- nls(
  y ~ mod_4(theta_0, theta_1, theta_2, x),
  start = list(
    theta_0 = as.numeric(gsub(".*=\\s*", "", theta_start[1])),
    theta_1 = as.numeric(gsub(".*=\\s*", "", theta_start[2])),
    theta_2 = as.numeric(gsub(".*=\\s*", "", theta_start[3]))
  ),
  data = my_data
)
summary(nlin_4)
#>
#> Formula: y ~ mod_4(theta_0, theta_1, theta_2, x)
#>
#> Parameters:
#>         Estimate Std. Error t value Pr(>|t|)
#> theta_0  3.63591    0.36528   9.954  < 2e-16 ***
#> theta_1 -1.30639    0.15561  -8.395 3.65e-15 ***
#> theta_2  0.50528    0.09215   5.483 1.03e-07 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
```

```
#> Residual standard error: 0.2831 on 247 degrees of freedom
#>
#> Number of iterations to convergence: 9
#> Achieved convergence tolerance: 2.294e-07
```

6.4.2 Logistic Growth Model

A classic logistic growth model follows the equation:

$$P = \frac{K}{1 + \exp(P_0 + rt)} + \epsilon$$

where:

- P = population at time t
- K = carrying capacity (maximum population)
- r = population growth rate
- P_0 = initial population log-ratio

However, R's built-in `SSlogis` function uses a slightly different parameterization:

$$P = \frac{asym}{1 + \exp\left(\frac{xmid - t}{scal}\right)}$$

where:

- $asym$ = carrying capacity (K)
- $xmid$ = the x-value at the inflection point of the curve
- $scal$ = scaling parameter

This gives the parameter relationships:

- $K = asym$
- $r = -1/scal$
- $P_0 = -r \cdot xmid$

```r
# Simulated time-series data
time <- c(1, 2, 3, 5, 10, 15, 20, 25, 30, 35)
population <- c(2.8, 4.2, 3.5, 6.3, 15.7, 21.3, 23.7, 25.1, 25.8, 25.9)

# Plot data points
plot(time, population, las = 1, pch = 16, main = "Logistic Growth Model")
```

```
# Fit the logistic growth model using programmed starting values
logisticModelSS <- nls(population ~ SSlogis(time, Asym, xmid, scal))

# Model summary
summary(logisticModelSS)
#>
#> Formula: population ~ SSlogis(time, Asym, xmid, scal)
#>
#> Parameters:
#>      Estimate Std. Error t value Pr(>|t|)
#> Asym  25.5029     0.3666   69.56 3.34e-11 ***
#> xmid   8.7347     0.3007   29.05 1.48e-08 ***
#> scal   3.6353     0.2186   16.63 6.96e-07 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.6528 on 7 degrees of freedom
#>
#> Number of iterations to convergence: 1
#> Achieved convergence tolerance: 1.908e-06

# Extract parameter estimates
coef(logisticModelSS)
#>      Asym      xmid      scal
#> 25.502890  8.734698  3.635333
```

To fit the model using an alternative parameterization (K, r, P_0), we convert

the estimated coefficients:

```
# Convert parameter estimates to alternative logistic model parameters
Ks <- as.numeric(coef(logisticModelSS)[1])  # Carrying capacity (K)
rs <- -1 / as.numeric(coef(logisticModelSS)[3])  # Growth rate (r)
Pos <- -rs * as.numeric(coef(logisticModelSS)[2])  # P_0

# Fit the logistic model with the alternative parameterization
logisticModel <- nls(
    population ~ K / (1 + exp(Po + r * time)),
    start = list(Po = Pos, r = rs, K = Ks)
)

# Model summary
summary(logisticModel)
#>
#> Formula: population ~ K/(1 + exp(Po + r * time))
#>
#> Parameters:
#>    Estimate Std. Error t value Pr(>|t|)
#> Po  2.40272    0.12702   18.92 2.87e-07 ***
#> r  -0.27508    0.01654  -16.63 6.96e-07 ***
#> K  25.50289    0.36665   69.56 3.34e-11 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.6528 on 7 degrees of freedom
#>
#> Number of iterations to convergence: 0
#> Achieved convergence tolerance: 1.924e-06
```

Visualizing the Logistic Model Fit

```
# Plot original data
plot(time,
     population,
     las = 1,
     pch = 16,
     main = "Logistic Growth Model Fit")

# Overlay the fitted logistic curve
lines(time,
      predict(logisticModel),
      col = "red",
      lwd = 2)
```

6.4.3 Nonlinear Plateau Model

This example is based on (Schabenberger and Pierce, 2001) and demonstrates the use of a **plateau model** to estimate the relationship between soil nitrate (NO_3) concentration and relative yield percent (RYP) at two different depths (30 cm and 60 cm).

```
# Load data
dat <- read.table("images/dat.txt", header = TRUE)

# Plot NO3 concentration vs. relative yield percent, colored by depth
library(ggplot2)
dat.plot <- ggplot(dat) +
  geom_point(aes(x = no3, y = ryp, color = as.factor(depth))) +
  labs(color = 'Depth (cm)') +
  xlab('Soil NO3 Concentration') +
  ylab('Relative Yield Percent') +
  theme_minimal()

# Display plot
dat.plot
```

The **suggested nonlinear plateau model** is given by:

$$E(Y_{ij}) = (\beta_{0j} + \beta_{1j} N_{ij}) I_{N_{ij} \leq \alpha_j} + (\beta_{0j} + \beta_{1j} \alpha_j) I_{N_{ij} > \alpha_j}$$

where:

- N_{ij} represents the soil nitrate (NO_3) concentration for observation i at depth j.
- i indexes individual observations.
- $j = 1, 2$ corresponds to depths **30 cm** and **60 cm**.

This model assumes a **linear increase** up to a threshold (α_j), beyond which the response **levels off (plateaus).**

Defining the Plateau Model as a Function

```
# Define the nonlinear plateau model function
nonlinModel <- function(predictor, b0, b1, alpha) {
  ifelse(predictor <= alpha,
         b0 + b1 * predictor,  # Linear growth below threshold
         b0 + b1 * alpha)      # Plateau beyond threshold
}
```

Creating a Self-Starting Function for `nls`

Since the model is **piecewise linear**, we can estimate starting values using:

1. A **linear regression** on the **first half** of sorted predictor values to estimate b_0 and b_1.

2. The **last predictor value** used in the regression as the plateau threshold (α)

```
# Define initialization function for self-starting plateau model
nonlinModelInit <- function(mCall, LHS, data) {
  # Sort data by increasing predictor value
  xy <- sortedXyData(mCall[['predictor']], LHS, data)
  n <- nrow(xy)

  # Fit a simple linear model using the first half of the sorted data
  lmFit <- lm(xy[1:(n / 2), 'y'] ~ xy[1:(n / 2), 'x'])

  # Extract initial estimates
  b0 <- coef(lmFit)[1]  # Intercept
  b1 <- coef(lmFit)[2]  # Slope
  alpha <- xy[(n / 2), 'x']  # Last x-value in the fitted linear range

  # Return initial parameter estimates
  value <- c(b0, b1, alpha)
  names(value) <- mCall[c('b0', 'b1', 'alpha')]
  value
}
```

Combining Model and Self-Start Function

```
# Define a self-starting nonlinear model for nls
SS_nonlinModel <- selfStart(nonlinModel,
                            nonlinModelInit,
                            c('b0', 'b1', 'alpha'))
```

The `nls` function is used to estimate parameters separately for **each soil depth (30 cm and 60 cm).**

```
# Fit the model for depth = 30 cm
sep30_nls <- nls(ryp ~ SS_nonlinModel(predictor = no3, b0, b1, alpha),
                 data = dat[dat$depth == 30,])

# Fit the model for depth = 60 cm
sep60_nls <- nls(ryp ~ SS_nonlinModel(predictor = no3, b0, b1, alpha),
                 data = dat[dat$depth == 60,])
```

We generate separate plots for **30 cm** and **60 cm** depths, showing both **confidence** and **prediction intervals.**

```
# Set plotting layout
par(mfrow = c(1, 2))

# Plot model fit for 30 cm depth
```

```r
plotFit(
  sep30_nls,
  interval = "both",
  pch = 19,
  shade = TRUE,
  col.conf = "skyblue4",
  col.pred = "lightskyblue2",
  data = dat[dat$depth == 30,],
  main = "Results at 30 cm Depth",
  ylab = "Relative Yield Percent",
  xlab = "Soil NO3 Concentration",
  xlim = c(0, 120)
)

# Plot model fit for 60 cm depth
plotFit(
  sep60_nls,
  interval = "both",
  pch = 19,
  shade = TRUE,
  col.conf = "lightpink4",
  col.pred = "lightpink2",
  data = dat[dat$depth == 60,],
  main = "Results at 60 cm Depth",
  ylab = "Relative Yield Percent",
  xlab = "Soil NO3 Concentration",
  xlim = c(0, 120)
)
```

```
summary(sep30_nls)
#>
#> Formula: ryp ~ SS_nonlinModel(predictor = no3, b0, b1, alpha)
#>
#> Parameters:
#>       Estimate Std. Error t value Pr(>|t|)
#> b0     15.1943     2.9781   5.102 6.89e-07 ***
#> b1      3.5760     0.1853  19.297  < 2e-16 ***
#> alpha  23.1324     0.5098  45.373  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 8.258 on 237 degrees of freedom
#>
#> Number of iterations to convergence: 6
#> Achieved convergence tolerance: 3.608e-09
summary(sep60_nls)
#>
#> Formula: ryp ~ SS_nonlinModel(predictor = no3, b0, b1, alpha)
#>
#> Parameters:
#>       Estimate Std. Error t value Pr(>|t|)
#> b0      5.4519     2.9785    1.83   0.0684 .
#> b1      5.6820     0.2529   22.46   <2e-16 ***
#> alpha  16.2863     0.2818   57.80   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#>
#> Residual standard error: 7.427 on 237 degrees of freedom
#>
#> Number of iterations to convergence: 5
#> Achieved convergence tolerance: 8.571e-09
```

Modeling Soil Depths Together and Comparing Models

Instead of fitting separate models for different soil depths, we first fit a **combined model** where all observations share a **common slope, intercept, and plateau**. We then test whether modeling the two depths separately provides a significantly better fit.

1. **Fitting a Reduced (Combined) Model**

The reduced model assumes that **all soil depths follow the same nonlinear relationship**.

```
# Fit the combined model (common parameters across all depths)
red_nls <- nls(
  ryp ~ SS_nonlinModel(predictor = no3, b0, b1, alpha),
  data = dat
)

# Display model summary
summary(red_nls)
#>
#> Formula: ryp ~ SS_nonlinModel(predictor = no3, b0, b1, alpha)
#>
#> Parameters:
#>       Estimate Std. Error t value Pr(>|t|)
#> b0      8.7901     2.7688   3.175   0.0016 **
#> b1      4.8995     0.2207  22.203   <2e-16 ***
#> alpha  18.0333     0.3242  55.630   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 9.13 on 477 degrees of freedom
#>
#> Number of iterations to convergence: 7
#> Achieved convergence tolerance: 7.126e-09

# Visualizing the combined model fit
par(mfrow = c(1, 1))
plotFit(
  red_nls,
  interval = "both",
```

```
  pch = 19,
  shade = TRUE,
  col.conf = "lightblue4",
  col.pred = "lightblue2",
  data = dat,
  main = 'Results for Combined Model',
  ylab = 'Relative Yield Percent',
  xlab = 'Soil NO3 Concentration'
)
```

2. **Examining Residuals for the Combined Model**

Checking residuals helps diagnose potential **lack of fit**.

```
library(nlstools)

# Residual diagnostics using nlstools
resid <- nlsResiduals(red_nls)

# Plot residuals
plot(resid)
```

If there is a **pattern** in the residuals (e.g., systematic deviations based on soil depth), this suggests that a **separate model for each depth** may be necessary.

3. **Testing Whether Depths Require Separate Models**

To formally test whether soil depth significantly affects the model parameters, we introduce a **parameterization where depth-specific parameters are increments from a baseline model** (30 cm depth):

$$\beta_{02} = \beta_{01} + d_0$$
$$\beta_{12} = \beta_{11} + d_1$$
$$\alpha_2 = \alpha_1 + d_a$$

where:

- $\beta_{01}, \beta_{11}, \alpha_1$ are parameters for **30 cm depth**.
- d_0, d_1, d_a represent **depth-specific differences** for **60 cm depth**.
- If d_0, d_1, d_a are significantly **different from 0**, the **two depths should be modeled separately**.

4. **Defining the Full (Depth-Specific) Model**

```
nonlinModelF <- function(predictor, soildep, b01, b11, a1, d0, d1, da) {

  # Define parameters for 60 cm depth as increments from 30 cm parameters
  b02 <- b01 + d0
  b12 <- b11 + d1
  a2 <- a1 + da
```

```
  # Compute model output for 30 cm depth
  y1 <- ifelse(
    predictor <= a1,
    b01 + b11 * predictor,
    b01 + b11 * a1
  )

  # Compute model output for 60 cm depth
  y2 <- ifelse(
    predictor <= a2,
    b02 + b12 * predictor,
    b02 + b12 * a2
  )

  # Assign correct model output based on depth
  y <- y1 * (soildep == 30) + y2 * (soildep == 60)

  return(y)
}
```

5. **Fitting the Full (Depth-Specific) Model**

The starting values are taken from the **separately fitted models** for each depth.

```
Soil_full <- nls(
  ryp ~ nonlinModelF(
    predictor = no3,
    soildep = depth,
    b01,
    b11,
    a1,
    d0,
    d1,
    da
  ),
  data = dat,
  start = list(
    b01 = 15.2,   # Intercept for 30 cm depth
    b11 = 3.58,   # Slope for 30 cm depth
    a1 = 23.13,   # Plateau cutoff for 30 cm depth
    d0 = -9.74,   # Intercept difference (60 cm - 30 cm)
    d1 = 2.11,    # Slope difference (60 cm - 30 cm)
    da = -6.85    # Plateau cutoff difference (60 cm - 30 cm)
  )
)
```

```r
# Display model summary
summary(Soil_full)
#>
#> Formula: ryp ~ nonlinModelF(predictor = no3, soildep = depth, b01, b11,
#>     a1, d0, d1, da)
#>
#> Parameters:
#>     Estimate Std. Error t value Pr(>|t|)
#> b01  15.1943     2.8322   5.365 1.27e-07 ***
#> b11   3.5760     0.1762  20.291  < 2e-16 ***
#> a1   23.1324     0.4848  47.711  < 2e-16 ***
#> d0   -9.7424     4.2357  -2.300   0.0219 *
#> d1    2.1060     0.3203   6.575 1.29e-10 ***
#> da   -6.8461     0.5691 -12.030  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 7.854 on 474 degrees of freedom
#>
#> Number of iterations to convergence: 1
#> Achieved convergence tolerance: 3.742e-06
```

6. **Model Comparison: Does Depth Matter?**

- If d_0, d_1, d_a **are significantly different from 0**, the depths should be modeled separately.
- The **p-values** for these parameters indicate whether depth-specific modeling is necessary.

Chapter 7

Generalized Linear Models

Generalized Linear Models (GLMs) extend the traditional linear regression framework to accommodate response variables that do not necessarily follow a normal distribution. They provide a **flexible approach** to modeling relationships between a set of predictors and various types of dependent variables.

While Ordinary Least Squares regression assumes that the response variable is continuous and normally distributed, GLMs allow for response variables that follow distributions from the **exponential family**, such as **binomial, Poisson, and gamma distributions**. This flexibility makes them particularly useful in a wide range of business and research applications.

A **Generalized Linear Model** (GLM) consists of three key components:

1. **A random component**: The response variable Y_i follows a distribution from the exponential family (e.g., binomial, Poisson, gamma).
2. **A systematic component**: A linear predictor $\eta_i = \mathbf{x_i'}\beta$, where $\mathbf{x_i'}$ is a vector of observed covariates (predictor variables) and β is a vector of parameters to be estimated.
3. **A link function**: A function $g(\cdot)$ that relates the expected value of the response variable, $\mu_i = E(Y_i)$, to the linear predictor (i.e., $\eta_i = g(\mu_i)$).

Although the relationship between the predictors and the outcome *may appear nonlinear* on the original outcome scale (due to the link function), a GLM is still considered "linear" in the statistical sense because it remains linear in the parameters β. Consequently, GLMs are **not** generally classified as nonlinear regression models. They "generalize" the traditional linear model by allowing for a broader range of response variable distributions and link functions, but retain linearity in their parameters.

The choice of **distribution** and **link function** depends on the nature of the response variable. In the following sections, we will explore several important GLM variants:

- Logistic Regression: Used for binary response variables (e.g., customer churn, loan defaults).
- Probit Regression: Similar to logistic regression but assumes a normal distribution for the underlying probability.
- Poisson Regression: Used for modeling count data (e.g., number of purchases, call center inquiries).
- Negative Binomial Regression: An extension of Poisson regression that accounts for overdispersion in count data.
- Quasi-Poisson Regression: A variation of Poisson regression that adjusts for overdispersion by allowing the variance to be a linear function of the mean.
- Multinomial Logistic Regression: A generalization of logistic regression for categorical response variables with more than two outcomes.
- Generalization of Generalized Linear Model: A flexible generalization of ordinary linear regression that allows for response variables with different distributions (e.g., normal, binomial, Poisson).

7.1 Logistic Regression

Logistic regression is a widely used Generalized Linear Model designed for modeling binary response variables. It is particularly useful in applications such as credit scoring, medical diagnosis, and customer churn prediction.

7.1.1 Logistic Model

Given a set of predictor variables $\mathbf{x}_i$, the probability of a positive outcome (e.g., success, event occurring) is modeled as:

$$p_i = f(\mathbf{x}_i; \beta) = \frac{\exp(\mathbf{x_i'}\ )}{1 + \exp(\mathbf{x_i'}\ )}$$

where:

- $p_i = \mathbb{E}[Y_i]$ is the probability of success for observation i.
- $\mathbf{x_i}$ is the vector of predictor variables.
- β is the vector of model coefficients.

7.1.1.1 Logit Transformation

The logistic function can be rewritten in terms of the **log-odds**, also known as the **logit function**:

$$\text{logit}(p_i) = \log\left(\frac{p_i}{1 - p_i}\right) = \mathbf{x_i'}$$

where:

- $\frac{p_i}{1-p_i}$ represents the **odds** of success (the ratio of the probability of success to the probability of failure).
- The logit function **ensures linearity** in the parameters, which aligns with the GLM framework.

Thus, logistic regression belongs to the family of **Generalized Linear Models** because **a function of the mean response (logit) is linear in the predictors**.

7.1.2 Likelihood Function

Since Y_i follows a **Bernoulli distribution** with probability p_i, the likelihood function for n independent observations is:

$$L(p_i) = \prod_{i=1}^{n} p_i^{Y_i}(1-p_i)^{1-Y_i}$$

By substituting the logistic function for p_i:

$$p_i = \frac{\exp(\mathbf{x_i'}\ )}{1+\exp(\mathbf{x_i'}\ )}, \quad 1-p_i = \frac{1}{1+\exp(\mathbf{x_i'}\ )}$$

we obtain:

$$L(\beta) = \prod_{i=1}^{n} \left(\frac{\exp(\mathbf{x_i'}\ )}{1+\exp(\mathbf{x_i'}\ )}\right)^{Y_i} \left(\frac{1}{1+\exp(\mathbf{x_i'}\ )}\right)^{1-Y_i}$$

Taking the natural logarithm of the likelihood function gives the **log-likelihood function**:

$$Q(\beta) = \log L(\beta) = \sum_{i=1}^{n} Y_i \mathbf{x_i'}\ - \sum_{i=1}^{n} \log(1+\exp(\mathbf{x_i'}\ ))$$

Since this function is **concave**, we can maximize it numerically using **iterative optimization techniques**, such as:

- **Newton-Raphson Method**
- **Fisher Scoring Algorithm**

These methods allow us to obtain the Maximum Likelihood Estimates of the parameters, $\hat{\beta}$.

Under standard regularity conditions, the **MLEs of logistic regression parameters are asymptotically normal**:

$$\hat{\beta} \dot{\sim} AN(\beta, [\mathbf{I}(\beta)]^{-1})$$

where:

- $\mathbf{I}(\beta)$ is the Fisher Information Matrix, which determines the variance-covariance structure of $\hat{\beta}$.

7.1.3 Fisher Information Matrix

The **Fisher Information Matrix** quantifies the amount of information that an observable random variable carries about the **unknown parameter** β. It is crucial in estimating the **variance-covariance matrix** of the estimated coefficients in logistic regression.

Mathematically, the Fisher Information Matrix is defined as:

$$\mathbf{I}(\beta) = E\left[\frac{\partial \log L(\beta)}{\partial \beta} \frac{\partial \log L(\beta)}{\partial \beta'}\right]$$

which expands to:

$$\mathbf{I}(\beta) = E\left[\left(\frac{\partial \log L(\beta)}{\partial \beta_i} \frac{\partial \log L(\beta)}{\partial \beta_j}\right)_{ij}\right]$$

Under **regularity conditions**, the Fisher Information Matrix is equivalent to the **negative expected Hessian matrix**:

$$\mathbf{I}(\beta) = -E\left[\frac{\partial^2 \log L(\beta)}{\partial \beta \partial \beta'}\right]$$

which further expands to:

$$\mathbf{I}(\beta) = -E\left[\left(\frac{\partial^2 \log L(\beta)}{\partial \beta_i \partial \beta_j}\right)_{ij}\right]$$

This representation is particularly useful because it allows us to compute the Fisher Information Matrix directly from the Hessian of the log-likelihood function.

Example: Fisher Information Matrix in Logistic Regression

Consider a **simple logistic regression model** with one predictor:

$$x_i'\beta = \beta_0 + \beta_1 x_i$$

From the log-likelihood function, the second-order partial derivatives are:

$$\begin{aligned}
-\frac{\partial^2 \ln(L(\beta))}{\partial \beta_0^2} &= \sum_{i=1}^{n} \frac{\exp(x_i'\beta)}{1+\exp(x_i'\beta)} - \left[\frac{\exp(x_i'\beta)}{1+\exp(x_i'\beta)}\right]^2 && \text{Intercept} \\
&= \sum_{i=1}^{n} p_i(1-p_i) \\
-\frac{\partial^2 \ln(L(\beta))}{\partial \beta_1^2} &= \sum_{i=1}^{n} \frac{x_i^2 \exp(x_i'\beta)}{1+\exp(x_i'\beta)} - \left[\frac{x_i \exp(x_i'\beta)}{1+\exp(x_i'\beta)}\right]^2 && \text{Slope} \\
&= \sum_{i=1}^{n} x_i^2 p_i(1-p_i) \\
-\frac{\partial^2 \ln(L(\beta))}{\partial \beta_0 \partial \beta_1} &= \sum_{i=1}^{n} \frac{x_i \exp(x_i'\beta)}{1+\exp(x_i'\beta)} - x_i \left[\frac{\exp(x_i'\beta)}{1+\exp(x_i'\beta)}\right]^2 && \text{Cross-derivative} \\
&= \sum_{i=1}^{n} x_i p_i(1-p_i)
\end{aligned}$$

Combining these elements, the **Fisher Information Matrix** for the logistic regression model is:

$$\mathbf{I}(\beta) = \begin{bmatrix} \sum_{i=1}^{n} p_i(1-p_i) & \sum_{i=1}^{n} x_i p_i(1-p_i) \\ \sum_{i=1}^{n} x_i p_i(1-p_i) & \sum_{i=1}^{n} x_i^2 p_i(1-p_i) \end{bmatrix}$$

where:

- $p_i = \frac{\exp(x_i'\beta)}{1+\exp(x_i'\beta)}$ represents the predicted probability.
- $p_i(1-p_i)$ is the **variance of the Bernoulli response variable**.
- The diagonal elements represent the variances of the estimated coefficients.
- The off-diagonal elements represent the covariances between β_0 and β_1.

The inverse of the Fisher Information Matrix provides the **variance-covariance matrix** of the estimated coefficients:

$$\mathbf{Var}(\hat{\beta}) = \mathbf{I}(\hat{\beta})^{-1}$$

This matrix is essential for:

- **Estimating standard errors** of the logistic regression coefficients.
- **Constructing confidence intervals** for β.
- **Performing hypothesis tests** (e.g., Wald Test).

```r
# Load necessary library
library(stats)

# Simulated dataset
set.seed(123)
n <- 100
x <- rnorm(n)
y <- rbinom(n, 1, prob = plogis(0.5 + 1.2 * x))

# Fit logistic regression model
model <- glm(y ~ x, family = binomial)

# Extract the Fisher Information Matrix (Negative Hessian)
fisher_info <- summary(model)$cov.unscaled

# Display the Fisher Information Matrix
print(fisher_info)
#>             (Intercept)          x
#> (Intercept)  0.05718171 0.01564322
#> x            0.01564322 0.10302992
```

7.1.4 Inference in Logistic Regression

Once we estimate the model parameters $\hat{\beta}$ using Maximum Likelihood Estimation, we can conduct inference to assess the significance of predictors, construct confidence intervals, and perform hypothesis testing. The two most common inference approaches in logistic regression are:

1. Likelihood Ratio Test
2. Wald Statistics

These tests rely on the **asymptotic normality** of MLEs and the properties of the Fisher Information Matrix.

7.1.4.1 Likelihood Ratio Test

The **Likelihood Ratio Test** compares two models:

- **Restricted Model**: A simpler model where some parameters are constrained to specific values.
- **Unrestricted Model**: The full model without constraints.

To test a hypothesis about a subset of parameters β_1, we leave β_2 (nuisance parameters) unspecified.

Hypothesis Setup:

$$H_0 : \beta_1 = \beta_{1,0}$$

where $\beta_{1,0}$ is a specified value (often zero). Let:

- $\hat{\beta}_{2,0}$ be the **MLE of** β_2 under the constraint $\beta_1 = \beta_{1,0}$.
- $\hat{\beta}_1, \hat{\beta}_2$ be the **MLEs under the full model**.

The **likelihood ratio test statistic** is:

$$-2 \log \Lambda = -2[\log L(\beta_{1,0}, \hat{\beta}_{2,0}) - \log L(\hat{\beta}_1, \hat{\beta}_2)]$$

where:

- The first term is the **log-likelihood of the restricted model**.
- The second term is the **log-likelihood of the unrestricted model**.

Under the null hypothesis:

$$-2 \log \Lambda \sim \chi^2_{\upsilon}$$

where υ is the number of restricted parameters. We **reject** H_0 if:

$$-2 \log \Lambda > \chi^2_{\upsilon, 1-\alpha}$$

Interpretation: If the likelihood ratio test statistic is large, this suggests that the restricted model (under H_0) fits significantly worse than the full model, leading us to reject the null hypothesis.

7.1.4.2 Wald Test

The **Wald test** is based on the asymptotic normality of MLEs:

$$\hat{\beta} \sim AN(\beta, [\mathbf{I}(\beta)]^{-1})$$

We test:

$$H_0 : \mathbf{L}\hat{\beta} = 0$$

where $\mathbf{L}$ is a $q \times p$ matrix with q linearly independent rows (often used to test multiple coefficients simultaneously). The **Wald test statistic** is:

$$W = (\mathbf{L}\hat{\boldsymbol{\beta}})'(\mathbf{L}[\mathbf{I}(\hat{\boldsymbol{\beta}})]^{-1}\mathbf{L}')^{-1}(\mathbf{L}\hat{\boldsymbol{\beta}})$$

Under H_0:

$$W \sim \chi^2_q$$

Interpretation: If W is large, the null hypothesis is rejected, suggesting that at least one of the tested coefficients is significantly different from zero.

Comparing Likelihood Ratio and Wald Tests

Test	Best Used When…
Likelihood Ratio Test	More accurate in small samples, providing better control of error rates. Recommended when sample sizes are small.
Wald Test	Easier to compute but may be inaccurate in small samples. Recommended when computational efficiency is a priority.

```r
# Load necessary library
library(stats)

# Simulate some binary outcome data
set.seed(123)
n <- 100
x <- rnorm(n)
y <- rbinom(n, 1, prob = plogis(0.5 + 1.2 * x))

# Fit logistic regression model
model <- glm(y ~ x, family = binomial)

# Display model summary (includes Wald tests)
summary(model)
#>
#> Call:
#> glm(formula = y ~ x, family = binomial)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -2.6226  -0.9385   0.5287   0.8333   1.4656
#>
```

```
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)   0.7223     0.2391   3.020 0.002524 **
#> x             1.2271     0.3210   3.823 0.000132 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 128.21  on 99  degrees of freedom
#> Residual deviance: 108.29  on 98  degrees of freedom
#> AIC: 112.29
#>
#> Number of Fisher Scoring iterations: 4

# Perform likelihood ratio test using anova()
anova(model, test="Chisq")
#> Analysis of Deviance Table
#>
#> Model: binomial, link: logit
#>
#> Response: y
#>
#> Terms added sequentially (first to last)
#>
#>
#>      Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
#> NULL                    99     128.21
#> x     1   19.913        98     108.29 8.105e-06 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

7.1.4.3 Confidence Intervals for Coefficients

A **95% confidence interval** for a logistic regression coefficient β_i is given by:

$$\hat{\beta}_i \pm 1.96\hat{s}_{ii}$$

where:

- $\hat{\beta}_i$ is the estimated coefficient.
- $\hat{s}_{ii}$ is the standard error (square root of the diagonal element of $[\mathbf{I}(\ )]^{-1}$).

This confidence interval provides a **range of plausible values** for β_i. If the interval does not include **zero**, we conclude that β_i is statistically significant.

- **For large sample sizes**, the Likelihood Ratio Test and Wald Test yield similar results.
- **For small sample sizes**, the Likelihood Ratio Test is preferred because the Wald test can be less reliable.

7.1.4.4 Interpretation of Logistic Regression Coefficients

For a **single predictor variable**, the logistic regression model is:

$$\text{logit}(\hat{p}_i) = \log\left(\frac{\hat{p}_i}{1-\hat{p}_i}\right) = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

where:

- $\hat{p}_i$ is the predicted probability of success at x_i.
- $\hat{\beta}_1$ represents the **log odds change** for a one-unit increase in x.

Interpreting β_1 in Terms of Odds

When the predictor variable increases by **one unit**, the logit of the probability changes by $\hat{\beta}_1$:

$$\text{logit}(\hat{p}_{x_i+1}) = \hat{\beta}_0 + \hat{\beta}_1(x_i+1) = \text{logit}(\hat{p}_{x_i}) + \hat{\beta}_1$$

Thus, the difference in log odds is:

$$\begin{aligned}\text{logit}(\hat{p}_{x_i+1}) - \text{logit}(\hat{p}_{x_i}) &= \log(\text{odds}(\hat{p}_{x_i+1})) - \log(\text{odds}(\hat{p}_{x_i})) \\ &= \log\left(\frac{\text{odds}(\hat{p}_{x_i+1})}{\text{odds}(\hat{p}_{x_i})}\right) \\ &= \hat{\beta}_1\end{aligned}$$

Exponentiating both sides:

$$\exp(\hat{\beta}_1) = \frac{\text{odds}(\hat{p}_{x_i+1})}{\text{odds}(\hat{p}_{x_i})}$$

This quantity, $\exp(\hat{\beta}_1)$, is the **odds ratio**, which quantifies the effect of a one-unit increase in x on the odds of success.

Generalization: Odds Ratio for Any Change in x

For a difference of c units in the predictor x, the estimated odds ratio is:

$$\exp(c\hat{\beta}_1)$$

For multiple predictors, $\exp(\hat{\beta}_k)$ represents the odds ratio for x_k, holding all other variables constant.

7.1.4.5 Inference on the Mean Response

For a given set of predictor values $x_h = (1, x_{h1}, ..., x_{h,p-1})'$, the estimated **mean response** (probability of success) is:

$$\hat{p}_h = \frac{\exp(\mathbf{x_h'}\hat{\ })}{1 + \exp(\mathbf{x_h'}\hat{\ })}$$

The **variance of the estimated probability** is:

$$s^2(\hat{p}_h) = \mathbf{x_h'}[\mathbf{I}(\hat{\ })]^{-1}\mathbf{x_h}$$

where:

- $\mathbf{I}(\hat{\beta})^{-1}$ is the **variance-covariance matrix** of $\hat{\beta}$.
- $s^2(\hat{p}_h)$ provides an estimate of **uncertainty** in $\hat{p}_h$.

In many applications, logistic regression is used for **classification**, where we predict whether an observation belongs to **category 0 or 1**. A commonly used decision rule is:

- Assign $y = 1$ if $\hat{p}_h \geq \tau$
- Assign $y = 0$ if $\hat{p}_h < \tau$

where τ is a chosen cutoff threshold (typically $\tau = 0.5$).

```r
# Load necessary library
library(stats)

# Simulated dataset
set.seed(123)
n <- 100
x <- rnorm(n)
y <- rbinom(n, 1, prob = plogis(0.5 + 1.2 * x))

# Fit logistic regression model
```

```r
model <- glm(y ~ x, family = binomial)

# Display model summary
summary(model)
#>
#> Call:
#> glm(formula = y ~ x, family = binomial)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -2.6226  -0.9385   0.5287   0.8333   1.4656
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)   0.7223     0.2391   3.020 0.002524 **
#> x             1.2271     0.3210   3.823 0.000132 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 128.21  on 99  degrees of freedom
#> Residual deviance: 108.29  on 98  degrees of freedom
#> AIC: 112.29
#>
#> Number of Fisher Scoring iterations: 4

# Extract coefficients and standard errors
coef_estimates <- coef(summary(model))
beta_hat <- coef_estimates[, 1]   # Estimated coefficients
se_beta  <- coef_estimates[, 2]   # Standard errors

# Compute 95% confidence intervals for coefficients
conf_intervals <- cbind(
  beta_hat - 1.96 * se_beta,
  beta_hat + 1.96 * se_beta
)

# Compute Odds Ratios
odds_ratios <- exp(beta_hat)

# Display results
print("Confidence Intervals for Coefficients:")
#> [1] "Confidence Intervals for Coefficients:"
print(conf_intervals)
```

```
#>                   [,1]     [,2]
#> (Intercept) 0.2535704 1.190948
#> x           0.5979658 1.856218

print("Odds Ratios:")
#> [1] "Odds Ratios:"
print(odds_ratios)
#> (Intercept)           x
#>    2.059080    3.411295

# Predict probability for a new observation (e.g., x = 1)
new_x <- data.frame(x = 1)
predicted_prob <- predict(model, newdata = new_x, type = "response")

print("Predicted Probability for x = 1:")
#> [1] "Predicted Probability for x = 1:"
print(predicted_prob)
#>         1
#> 0.8753759
```

7.1.5 Application: Logistic Regression

In this section, we demonstrate the application of **logistic regression** using simulated data. We explore model fitting, inference, residual analysis, and goodness-of-fit testing.

1. Load Required Libraries

```
library(kableExtra)
library(dplyr)
library(pscl)
library(ggplot2)
library(faraway)
library(nnet)
library(agridat)
library(nlstools)
```

2. Data Generation

We generate a dataset where the predictor variable X follows a **uniform distribution**:

$$x \sim Unif(-0.5, 2.5)$$

The **linear predictor** is given by:

$$\eta = 0.5 + 0.75x$$

Passing η into the inverse-logit function, we obtain:

$$p = \frac{\exp(\eta)}{1 + \exp(\eta)}$$

which ensures that $p \in [0, 1]$. We then generate the binary response variable:

$$y \sim Bernoulli(p)$$

```
set.seed(23) # Set seed for reproducibility
x <- runif(1000, min = -0.5, max = 2.5)  # Generate X values
eta1 <- 0.5 + 0.75 * x                   # Compute linear predictor
p <- exp(eta1) / (1 + exp(eta1))          # Compute probabilities
y <- rbinom(1000, 1, p)                   # Generate binary response
BinData <- data.frame(X = x, Y = y)       # Create data frame
```

3. Model Fitting

We fit a **logistic regression model** to the simulated data:

$$\text{logit}(p) = \beta_0 + \beta_1 X$$

```
Logistic_Model <- glm(formula = Y ~ X,
                      # Specifies the response distribution
                      family = binomial,
                      data = BinData)

summary(Logistic_Model) # Model summary
#>
#> Call:
#> glm(formula = Y ~ X, family = binomial, data = BinData)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -2.2317   0.4153   0.5574   0.7922   1.1469
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  0.46205    0.10201   4.530 5.91e-06 ***
#> X            0.78527    0.09296   8.447  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 1106.7  on 999  degrees of freedom
#> Residual deviance: 1027.4  on 998  degrees of freedom
#> AIC: 1031.4
#>
#> Number of Fisher Scoring iterations: 4
nlstools::confint2(Logistic_Model) # Confidence intervals
#>                 2.5 %    97.5 %
#> (Intercept) 0.2618709 0.6622204
#> X           0.6028433 0.9676934

# Compute odds ratios
OddsRatio <- coef(Logistic_Model) %>% exp
OddsRatio
#> (Intercept)           X
#>    1.587318    2.192995
```

Interpretation of the Odds Ratio

- When $x = 0$, the **odds of success** are **1.59**.
- When $x = 1$, the **odds of success increase by a factor of 2.19**, indicating a **119.29% increase**.

4. Deviance Test

We assess the model's significance using the **deviance test**, which compares:

- H_0: No predictors are related to the response (intercept-only model).
- H_1: At least one predictor is related to the response.

The **test statistic** is:

$$D = \text{Null Deviance} - \text{Residual Deviance}$$

```
Test_Dev <- Logistic_Model$null.deviance - Logistic_Model$deviance
p_val_dev <- 1 - pchisq(q = Test_Dev, df = 1)
p_val_dev
#> [1] 0
```

Conclusion:

Since the **p-value is approximately 0**, we **reject** H_0, confirming that X is significantly related to Y.

5. Residual Analysis

We compute **deviance residuals** and plot them against X.

```
Logistic_Resids <- residuals(Logistic_Model, type = "deviance")

plot(
    y = Logistic_Resids,
    x = BinData$X,
    xlab = 'X',
    ylab = 'Deviance Residuals'
)
```

This plot is not very informative. A more insightful approach is **binned residual plots**.

6. Binned Residual Plot

We group residuals into **bins** based on predicted values.

```
plot_bin <- function(Y,
                     X,
                     bins = 100,
                     return.DF = FALSE) {
  Y_Name <- deparse(substitute(Y))
  X_Name <- deparse(substitute(X))

  Binned_Plot <- data.frame(Plot_Y = Y, Plot_X = X)
  Binned_Plot$bin <-
    cut(Binned_Plot$Plot_X, breaks = bins) %>% as.numeric

  Binned_Plot_summary <- Binned_Plot %>%
```

```
    group_by(bin) %>%
    summarise(
      Y_ave = mean(Plot_Y),
      X_ave = mean(Plot_X),
      Count = n()
    ) %>% as.data.frame

  plot(
    y = Binned_Plot_summary$Y_ave,
    x = Binned_Plot_summary$X_ave,
    ylab = Y_Name,
    xlab = X_Name
  )

  if (return.DF)
    return(Binned_Plot_summary)
}

plot_bin(Y = Logistic_Resids, X = BinData$X, bins = 100)
```

We also examine **predicted values vs residuals**:

```
Logistic_Predictions <- predict(Logistic_Model, type = "response")
plot_bin(Y = Logistic_Resids, X = Logistic_Predictions, bins = 100)
```

Finally, we compare **predicted probabilities** to actual outcomes:

```
NumBins <- 10
Binned_Data <- plot_bin(
    Y = BinData$Y,
    X = Logistic_Predictions,
    bins = NumBins,
    return.DF = TRUE
)
Binned_Data
#>    bin     Y_ave     X_ave Count
#> 1    1 0.5833333 0.5382095    72
#> 2    2 0.5200000 0.5795887    75
#> 3    3 0.6567164 0.6156540    67
#> 4    4 0.7014925 0.6579674    67
#> 5    5 0.6373626 0.6984765    91
#> 6    6 0.7500000 0.7373341    72
#> 7    7 0.7096774 0.7786747    93
#> 8    8 0.8503937 0.8203819   127
#> 9    9 0.8947368 0.8601232   133
#> 10  10 0.8916256 0.9004734   203
abline(0, 1, lty = 2, col = 'blue')
```

7. Model Goodness-of-Fit: Hosmer-Lemeshow Test

The **Hosmer-Lemeshow test** evaluates whether the model fits the data well. The test statistic is:

$$X_{HL}^2 = \sum_{j=1}^{J} \frac{(y_j - m_j \hat{p}_j)^2}{m_j \hat{p}_j (1 - \hat{p}_j)}$$

where:

- y_j is the observed number of successes in bin j.
- m_j is the number of observations in bin j.
- $\hat{p}_j$ is the predicted probability in bin j.

Under H_0, we assume:

$$X_{HL}^2 \sim \chi_{J-1}^2$$

```
HL_BinVals <- (Binned_Data$Count * Binned_Data$Y_ave -
               Binned_Data$Count * Binned_Data$X_ave) ^ 2 /
               (Binned_Data$Count * Binned_Data$X_ave * (1 - Binned_Data$X_ave))

HLpval <- pchisq(q = sum(HL_BinVals),
                 df = NumBins - 1,
                 lower.tail = FALSE)
HLpval
#> [1] 0.4150004
```

Conclusion:

- Since p-value $= 0.99$, we **fail to reject** H_0.
- This indicates that **the model fits the data well**.

7.2 Probit Regression

Probit regression is a type of Generalized Linear Models used for binary outcome variables. Unlike logistic regression, which uses the logit function, probit regression assumes that the probability of success is determined by an **underlying normally distributed latent variable**.

7.2.1 Probit Model

Let Y_i be a binary response variable:

$$Y_i = \begin{cases} 1, & \text{if success occurs} \\ 0, & \text{otherwise} \end{cases}$$

We assume that Y_i follows a **Bernoulli distribution**:

$$Y_i \sim \text{Bernoulli}(p_i), \quad \text{where } p_i = P(Y_i = 1|\mathbf{x_i})$$

Instead of the logit function in logistic regression:

$$\text{logit}(p_i) = \log\left(\frac{p_i}{1-p_i}\right) = \mathbf{x_i'}$$

Probit regression uses the **inverse standard normal CDF**:

$$\Phi^{-1}(p_i) = \mathbf{x_i'}$$

where:

- $\Phi(\cdot)$ is the **CDF of the standard normal distribution**.
- $\mathbf{x_i}$ is the vector of predictors.
- θ is the vector of regression coefficients.

Thus, the probability of success is:

$$p_i = P(Y_i = 1|\mathbf{x_i}) = \Phi(\mathbf{x_i'}\ )$$

where:

$$\Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt$$

7.2.2 Application: Probit Regression

```
# Load necessary library
library(ggplot2)

# Set seed for reproducibility
set.seed(123)

# Simulate data
n <- 1000
x1 <- rnorm(n)
x2 <- rnorm(n)
latent <- 0.5 * x1 + 0.7 * x2 + rnorm(n)  # Linear combination
y <- ifelse(latent > 0, 1, 0)  # Binary outcome

# Create dataframe
data <- data.frame(y, x1, x2)

# Fit Probit model
probit_model <-
    glm(y ~ x1 + x2, family = binomial(link = "probit"), data = data)
summary(probit_model)
#>
#> Call:
#> glm(formula = y ~ x1 + x2, family = binomial(link = "probit"),
#>     data = data)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -2.3740  -0.8663  -0.2318   0.8684   2.6666
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept) -0.09781    0.04499  -2.174   0.0297 *
#> x1           0.43838    0.04891   8.963   <2e-16 ***
#> x2           0.75538    0.05306  14.235   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
```

```
#>     Null deviance: 1385.1  on 999  degrees of freedom
#> Residual deviance: 1045.3  on 997  degrees of freedom
#> AIC: 1051.3
#>
#> Number of Fisher Scoring iterations: 5

# Fit Logit model
logit_model <-
    glm(y ~ x1 + x2, family = binomial(link = "logit"), data = data)
summary(logit_model)
#>
#> Call:
#> glm(formula = y ~ x1 + x2, family = binomial(link = "logit"),
#>     data = data)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -2.3048  -0.8571  -0.2805   0.8632   2.5335
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept) -0.16562    0.07600  -2.179   0.0293 *
#> x1           0.73234    0.08507   8.608   <2e-16 ***
#> x2           1.25220    0.09486  13.201   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 1385.1  on 999  degrees of freedom
#> Residual deviance: 1048.4  on 997  degrees of freedom
#> AIC: 1054.4
#>
#> Number of Fisher Scoring iterations: 4

# Compare Coefficients
coef_comparison <- data.frame(
    Variable = names(coef(probit_model)),
    Probit_Coef = coef(probit_model),
    Logit_Coef = coef(logit_model),
    Logit_Probit_Ratio = coef(logit_model) / coef(probit_model)
)

print(coef_comparison)
#>                Variable Probit_Coef Logit_Coef Logit_Probit_Ratio
```

```
#> (Intercept) (Intercept) -0.09780689 -0.1656216          1.693353
#> x1                   x1  0.43837627  0.7323392          1.670572
#> x2                   x2  0.75538259  1.2522008          1.657704

# Compute predicted probabilities
data$probit_pred <- predict(probit_model, type = "response")
data$logit_pred <- predict(logit_model, type = "response")

# Plot Probit vs Logit predictions
ggplot(data, aes(x = probit_pred, y = logit_pred)) +
    geom_point(alpha = 0.5) +
    geom_abline(slope = 1,
                intercept = 0,
                col = "red") +
    labs(title = "Comparison of Predicted Probabilities",
         x = "Probit Predictions", y = "Logit Predictions")
```

```
# Classification Accuracy
threshold <- 0.5
data$probit_class <- ifelse(data$probit_pred > threshold, 1, 0)
data$logit_class <- ifelse(data$logit_pred > threshold, 1, 0)

probit_acc <- mean(data$probit_class == data$y)
logit_acc <- mean(data$logit_class == data$y)

print(paste("Probit Accuracy:", round(probit_acc, 4)))
```

```
#> [1] "Probit Accuracy: 0.71"
print(paste("Logit Accuracy:", round(logit_acc, 4)))
#> [1] "Logit Accuracy: 0.71"
```

7.3 Binomial Regression

In previous sections, we introduced **binomial regression models**, including both Logistic Regression and probit regression, and discussed their theoretical foundations. Now, we apply these methods to real-world data using the `esoph` dataset, which examines the relationship between esophageal cancer and potential risk factors such as **alcohol consumption** and **age group**.

7.3.1 Dataset Overview

The `esoph` dataset consists of:

- **Successes (`ncases`)**: The number of individuals diagnosed with esophageal cancer.
- **Failures (`ncontrols`)**: The number of individuals in the control group (without cancer).
- **Predictors**:
  - `agegp`: Age group of individuals.
  - `alcgp`: Alcohol consumption category.
  - `tobgp`: Tobacco consumption category.

Before fitting our models, let's inspect the dataset and visualize some key relationships.

```
# Load and inspect the dataset
data("esoph")
head(esoph, n = 3)
#>   agegp     alcgp    tobgp ncases ncontrols
#> 1 25-34 0-39g/day 0-9g/day      0        40
#> 2 25-34 0-39g/day    10-19      0        10
#> 3 25-34 0-39g/day    20-29      0         6

# Visualizing the proportion of cancer cases by alcohol consumption
plot(
  esoph$ncases / (esoph$ncases + esoph$ncontrols) ~ esoph$alcgp,
  ylab = "Proportion of Cancer Cases",
  xlab = "Alcohol Consumption Group",
  main = "Esophageal Cancer Data"
)
```

```
# Ensure categorical variables are treated as factors
class(esoph$agegp) <- "factor"
class(esoph$alcgp) <- "factor"
class(esoph$tobgp) <- "factor"
```

7.3.2 Apply Logistic Model

We first fit a **logistic regression model**, where the response variable is the proportion of cancer cases (`ncases`) relative to total observations (`ncases + ncontrols`).

```
# Logistic regression using alcohol consumption as a predictor
model <-
    glm(cbind(ncases, ncontrols) ~ alcgp,
        data = esoph,
        family = binomial)

# Summary of the model
summary(model)
#>
#> Call:
#> glm(formula = cbind(ncases, ncontrols) ~ alcgp, family = binomial,
#>     data = esoph)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -4.0759  -1.2037  -0.0183   1.0928   3.7336
```

```
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  -2.5885     0.1925 -13.444  < 2e-16 ***
#> alcgp40-79    1.2712     0.2323   5.472 4.46e-08 ***
#> alcgp80-119   2.0545     0.2611   7.868 3.59e-15 ***
#> alcgp120+     3.3042     0.3237  10.209  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 367.95  on 87  degrees of freedom
#> Residual deviance: 221.46  on 84  degrees of freedom
#> AIC: 344.51
#>
#> Number of Fisher Scoring iterations: 5
```

Interpretation

- The coefficients represent the **log-odds** of having esophageal cancer relative to the baseline alcohol consumption group.
- **P-values** indicate whether alcohol consumption levels significantly influence cancer risk.

Model Diagnostics

```
# Convert coefficients to odds ratios
exp(coefficients(model))
#> (Intercept)  alcgp40-79 alcgp80-119   alcgp120+
#>  0.07512953  3.56527094  7.80261593 27.22570533

# Model goodness-of-fit measures
deviance(model) / df.residual(model)  # Closer to 1 suggests a better fit
#> [1] 2.63638
model$aic  # Lower AIC is preferable for model comparison
#> [1] 344.5109
```

To improve our model, we include **age group (`agegp`)** as an additional predictor.

```
# Logistic regression with alcohol consumption and age
better_model <- glm(
    cbind(ncases, ncontrols) ~ agegp + alcgp,
    data = esoph,
    family = binomial
)
```

```r
# Summary of the improved model
summary(better_model)
#>
#> Call:
#> glm(formula = cbind(ncases, ncontrols) ~ agegp + alcgp, family = binomial,
#>     data = esoph)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -2.2395  -0.7186  -0.2324   0.7930   3.3538
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  -6.1472     1.0419  -5.900 3.63e-09 ***
#> agegp35-44    1.6311     1.0800   1.510 0.130973
#> agegp45-54    3.4258     1.0389   3.297 0.000976 ***
#> agegp55-64    3.9435     1.0346   3.811 0.000138 ***
#> agegp65-74    4.3568     1.0413   4.184 2.87e-05 ***
#> agegp75+      4.4242     1.0914   4.054 5.04e-05 ***
#> alcgp40-79    1.4343     0.2448   5.859 4.64e-09 ***
#> alcgp80-119   2.0071     0.2776   7.230 4.84e-13 ***
#> alcgp120+     3.6800     0.3763   9.778  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 367.95  on 87  degrees of freedom
#> Residual deviance: 105.88  on 79  degrees of freedom
#> AIC: 238.94
#>
#> Number of Fisher Scoring iterations: 6

# Model evaluation
better_model$aic # Lower AIC is better
#> [1] 238.9361

# Convert coefficients to odds ratios
# exp(coefficients(better_model))
data.frame(`Odds Ratios` = exp(coefficients(better_model)))
#>              Odds.Ratios
#> (Intercept)  0.002139482
#> agegp35-44   5.109601844
#> agegp45-54  30.748594216
#> agegp55-64  51.596634690
```

```
#> agegp65-74  78.005283850
#> agegp75+    83.448437749
#> alcgp40-79   4.196747169
#> alcgp80-119  7.441782227
#> alcgp120+   39.646885126

# Compare models using likelihood ratio test (Chi-square test)
pchisq(
    q = model$deviance - better_model$deviance,
    df = model$df.residual - better_model$df.residual,
    lower.tail = FALSE
)
#> [1] 2.713923e-23
```

Key Takeaways

- **AIC Reduction**: A lower AIC suggests that adding age as a predictor improves the model.
- **Likelihood Ratio Test**: This test compares the two models and determines whether the improvement is statistically significant.

7.3.3 Apply Probit Model

As discussed earlier, the **probit** model is an alternative to logistic regression, using a cumulative normal distribution instead of the logistic function.

```
# Probit regression model
Prob_better_model <- glm(
    cbind(ncases, ncontrols) ~ agegp + alcgp,
    data = esoph,
    family = binomial(link = probit)
)

# Summary of the probit model
summary(Prob_better_model)
#>
#> Call:
#> glm(formula = cbind(ncases, ncontrols) ~ agegp + alcgp, family = binomial(link = pr
#>     data = esoph)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -2.1325  -0.6877  -0.1661   0.7654   3.3258
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
```

```
#> (Intercept)  -3.3741     0.4922  -6.855 7.13e-12 ***
#> agegp35-44    0.8562     0.5081   1.685 0.092003 .
#> agegp45-54    1.7829     0.4904   3.636 0.000277 ***
#> agegp55-64    2.1034     0.4876   4.314 1.61e-05 ***
#> agegp65-74    2.3374     0.4930   4.741 2.13e-06 ***
#> agegp75+      2.3694     0.5275   4.491 7.08e-06 ***
#> alcgp40-79    0.8080     0.1330   6.076 1.23e-09 ***
#> alcgp80-119   1.1399     0.1558   7.318 2.52e-13 ***
#> alcgp120+     2.1204     0.2060  10.295  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 367.95  on 87  degrees of freedom
#> Residual deviance: 104.48  on 79  degrees of freedom
#> AIC: 237.53
#>
#> Number of Fisher Scoring iterations: 6
```

Why Consider a Probit Model?

- Like logistic regression, probit regression estimates probabilities, but it assumes a **normal distribution of the latent variable**.
- While the **interpretation of coefficients differs**, model comparisons can still be made using **AIC**.

7.4 Poisson Regression

7.4.1 The Poisson Distribution

Poisson regression is used for modeling **count data**, where the response variable represents the number of occurrences of an event within a fixed period, space, or other unit. The Poisson distribution is defined as:

$$
\begin{aligned}
f(Y_i) &= \frac{\mu_i^{Y_i} \exp(-\mu_i)}{Y_i!}, \quad Y_i = 0, 1, 2, ... \\
E(Y_i) &= \mu_i \\
\text{Var}(Y_i) &= \mu_i
\end{aligned}
$$

where:

- Y_i is the count variable.
- μ_i is the expected count for the i-th observation.

- The **mean and variance are equal** $E(Y_i) = \text{Var}(Y_i)$, making Poisson regression suitable when variance follows this property.

However, real-world count data often exhibit **overdispersion**, where the variance exceeds the mean. We will discuss remedies such as Quasi-Poisson and Negative Binomial Regression later.

7.4.2 Poisson Model

We model the expected count μ_i as a function of predictors $\mathbf{x_i}$ and parameters θ:

$$\mu_i = f(\mathbf{x_i};\ )$$

7.4.3 Link Function Choices

Since μ_i must be positive, we often use a **log-link function**:

$$\theta \log(\mu_i) = \mathbf{x_i'}$$

This ensures that the predicted counts are always non-negative. This is analogous to logistic regression, where we use the logit link for binary outcomes.

Rewriting:

$$\mu_i = \exp(\mathbf{x_i'}\ )$$

which ensures $\mu_i > 0$ for all parameter values.

7.4.4 Application: Poisson Regression

We apply Poisson regression to the **bioChemists** dataset (from the `pscl` package), which contains information on academic productivity in terms of published articles.

7.4.4.1 Dataset Overview

```
library(tidyverse)
# Load dataset
data(bioChemists, package = "pscl")

# Rename columns for clarity
bioChemists <- bioChemists %>%
  rename(
    Num_Article = art,
    # Number of articles in last 3 years
```

```
    Sex = fem,
    # 1 if female, 0 if male
    Married = mar,
    # 1 if married, 0 otherwise
    Num_Kid5 = kid5,
    # Number of children under age 6
    PhD_Quality = phd,
    # Prestige of PhD program
    Num_MentArticle = ment   # Number of articles by mentor in last 3 years
  )

# Visualize response variable distribution
hist(bioChemists$Num_Article,
     breaks = 25,
     main = "Number of Articles Published")
```

The **distribution of the number of articles** is right-skewed, which suggests a Poisson model may be appropriate.

7.4.4.2 Fitting a Poisson Regression Model

We model the number of articles published (`Num_Article`) as a function of various predictors.

```
# Poisson regression model
Poisson_Mod <-
  glm(Num_Article ~ .,
      family = poisson,
```

```
      data = bioChemists)

# Summary of the model
summary(Poisson_Mod)
#>
#> Call:
#> glm(formula = Num_Article ~ ., family = poisson, data = bioChemists)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -3.5672  -1.5398  -0.3660   0.5722   5.4467
#>
#> Coefficients:
#>                  Estimate Std. Error z value Pr(>|z|)
#> (Intercept)      0.304617   0.102981   2.958   0.0031 **
#> SexWomen        -0.224594   0.054613  -4.112 3.92e-05 ***
#> MarriedMarried   0.155243   0.061374   2.529   0.0114 *
#> Num_Kid5        -0.184883   0.040127  -4.607 4.08e-06 ***
#> PhD_Quality      0.012823   0.026397   0.486   0.6271
#> Num_MentArticle  0.025543   0.002006  12.733  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for poisson family taken to be 1)
#>
#>     Null deviance: 1817.4  on 914  degrees of freedom
#> Residual deviance: 1634.4  on 909  degrees of freedom
#> AIC: 3314.1
#>
#> Number of Fisher Scoring iterations: 5
```

Interpretation:

- **Coefficients** are on the **log scale**, meaning they represent **log-rate ratios**.
- **Exponentiating** the coefficients gives the **rate ratios**.
- **Statistical significance** tells us whether each variable has a meaningful impact on publication count.

7.4.4.3 Model Diagnostics: Goodness of Fit

7.4.4.3.1 Pearson's Chi-Square Test for Overdispersion We compute the **Pearson chi-square statistic** to check whether the variance significantly

exceeds the mean.

$$X^2 = \sum \frac{(Y_i - \hat{\mu}_i)^2}{\hat{\mu}_i}$$

```
# Compute predicted means
Predicted_Means <- predict(Poisson_Mod, type = "response")

# Pearson chi-square test
X2 <-
  sum((bioChemists$Num_Article - Predicted_Means) ^ 2 / Predicted_Means)
X2
#> [1] 1662.547
pchisq(X2, Poisson_Mod$df.residual, lower.tail = FALSE)
#> [1] 7.849882e-47
```

- If **p-value is small**, overdispersion is present.
- Large **X^2 statistic** suggests the model may not adequately capture variability.

7.4.4.3.2 Overdispersion Check: Ratio of Deviance to Degrees of Freedom We compute:

$$\hat{\phi} = \frac{\text{deviance}}{\text{degrees of freedom}}$$

```
# Overdispersion check
Poisson_Mod$deviance / Poisson_Mod$df.residual
#> [1] 1.797988
```

- If $\hat{\phi} > 1$, overdispersion is likely present.
- A value significantly above 1 suggests the need for an alternative model.

7.4.4.4 Addressing Overdispersion

7.4.4.4.1 Including Interaction Terms One possible remedy is to incorporate **interaction terms**, capturing complex relationships between predictors.

```
# Adding two-way and three-way interaction terms
Poisson_Mod_All2way <-
  glm(Num_Article ~ . ^ 2, family = poisson, data = bioChemists)
Poisson_Mod_All3way <-
  glm(Num_Article ~ . ^ 3, family = poisson, data = bioChemists)
```

This may improve model fit, but can lead to overfitting.

7.4.4.4.2 Quasi-Poisson Model (Adjusting for Overdispersion) A quick fix is to **allow the variance to scale** by introducing $\hat{\phi}$:

$$\text{Var}(Y_i) = \hat{\phi}\mu_i$$

```
# Estimate dispersion parameter
phi_hat = Poisson_Mod$deviance / Poisson_Mod$df.residual

# Adjusting Poisson model to account for overdispersion
summary(Poisson_Mod, dispersion = phi_hat)
#>
#> Call:
#> glm(formula = Num_Article ~ ., family = poisson, data = bioChemists)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -3.5672  -1.5398  -0.3660   0.5722   5.4467
#>
#> Coefficients:
#>                 Estimate Std. Error z value Pr(>|z|)
#> (Intercept)      0.30462    0.13809   2.206  0.02739 *
#> SexWomen        -0.22459    0.07323  -3.067  0.00216 **
#> MarriedMarried   0.15524    0.08230   1.886  0.05924 .
#> Num_Kid5        -0.18488    0.05381  -3.436  0.00059 ***
#> PhD_Quality      0.01282    0.03540   0.362  0.71715
#> Num_MentArticle  0.02554    0.00269   9.496  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for poisson family taken to be 1.797988)
#>
#>     Null deviance: 1817.4  on 914  degrees of freedom
#> Residual deviance: 1634.4  on 909  degrees of freedom
#> AIC: 3314.1
#>
#> Number of Fisher Scoring iterations: 5
```

Alternatively, we refit using a **Quasi-Poisson model**, which adjusts standard errors:

```
# Quasi-Poisson model
quasiPoisson_Mod <- glm(Num_Article ~ ., family = quasipoisson, data = bioChemists)
summary(quasiPoisson_Mod)
#>
#> Call:
#> glm(formula = Num_Article ~ ., family = quasipoisson, data = bioChemists)
```

```
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -3.5672  -1.5398  -0.3660   0.5722   5.4467
#>
#> Coefficients:
#>                  Estimate Std. Error t value Pr(>|t|)
#> (Intercept)      0.304617   0.139273   2.187 0.028983 *
#> SexWomen        -0.224594   0.073860  -3.041 0.002427 **
#> MarriedMarried   0.155243   0.083003   1.870 0.061759 .
#> Num_Kid5        -0.184883   0.054268  -3.407 0.000686 ***
#> PhD_Quality      0.012823   0.035700   0.359 0.719544
#> Num_MentArticle  0.025543   0.002713   9.415  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for quasipoisson family taken to be 1.829006)
#>
#>     Null deviance: 1817.4  on 914  degrees of freedom
#> Residual deviance: 1634.4  on 909  degrees of freedom
#> AIC: NA
#>
#> Number of Fisher Scoring iterations: 5
```

While Quasi-Poisson corrects standard errors, **it does not introduce an extra parameter** for overdispersion.

7.4.4.4.3 Negative Binomial Regression (Preferred Approach) A Negative Binomial Regression explicitly models overdispersion by introducing a **dispersion parameter** θ:

$$\text{Var}(Y_i) = \mu_i + \theta\mu_i^2$$

This extends Poisson regression by allowing the variance to **grow quadratically** rather than linearly.

```r
# Load MASS package
library(MASS)

# Fit Negative Binomial regression
NegBin_Mod <- glm.nb(Num_Article ~ ., data = bioChemists)

# Model summary
summary(NegBin_Mod)
#>
```

```
#> Call:
#> glm.nb(formula = Num_Article ~ ., data = bioChemists, init.theta = 2.264387695,
#>     link = log)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -2.1678  -1.3617  -0.2806   0.4476   3.4524
#>
#> Coefficients:
#>                  Estimate Std. Error z value Pr(>|z|)
#> (Intercept)      0.256144   0.137348   1.865 0.062191 .
#> SexWomen        -0.216418   0.072636  -2.979 0.002887 **
#> MarriedMarried   0.150489   0.082097   1.833 0.066791 .
#> Num_Kid5        -0.176415   0.052813  -3.340 0.000837 ***
#> PhD_Quality      0.015271   0.035873   0.426 0.670326
#> Num_MentArticle  0.029082   0.003214   9.048  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for Negative Binomial(2.2644) family taken to be 1)
#>
#>     Null deviance: 1109.0  on 914  degrees of freedom
#> Residual deviance: 1004.3  on 909  degrees of freedom
#> AIC: 3135.9
#>
#> Number of Fisher Scoring iterations: 1
#>
#>
#>               Theta:  2.264
#>           Std. Err.:  0.271
#>
#>  2 x log-likelihood:  -3121.917
```

This model is generally **preferred over Quasi-Poisson**, as it explicitly accounts for **heterogeneity** in the data.

7.5 Negative Binomial Regression

When modeling **count data**, Poisson regression assumes that the **mean and variance are equal**:

$$\text{Var}(Y_i) = E(Y_i) = \mu_i$$

However, in many real-world datasets, the variance exceeds the mean—a phenomenon known as **overdispersion**. When overdispersion is present, the **Pois-**

son model underestimates the variance, leading to:

- **Inflated test statistics** (small p-values).
- **Overconfident predictions**.
- **Poor model fit**.

7.5.1 Negative Binomial Distribution

To address overdispersion, **Negative Binomial (NB) regression** introduces an extra **dispersion parameter** θ to allow variance to be greater than the mean:

$$\text{Var}(Y_i) = \mu_i + \theta\mu_i^2$$

where:

- $\mu_i = \exp(\mathbf{x_i'}\ )$ is the expected count.
- θ is the dispersion parameter.
- When $\theta \to 0$, the NB model reduces to the Poisson model.

Thus, Negative Binomial regression is a **generalization of Poisson regression** that accounts for overdispersion.

7.5.2 Application: Negative Binomial Regression

We apply **Negative Binomial regression** to the `bioChemists` dataset to model the number of research articles (`Num_Article`) as a function of several predictors.

7.5.2.1 Fitting the Negative Binomial Model

```
# Load necessary package
library(MASS)

# Fit Negative Binomial model
NegBinom_Mod <- MASS::glm.nb(Num_Article ~ ., data = bioChemists)

# Model summary
summary(NegBinom_Mod)
#>
#> Call:
#> MASS::glm.nb(formula = Num_Article ~ ., data = bioChemists, init.theta = 2.264387695,
#>     link = log)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -2.1678  -1.3617  -0.2806   0.4476   3.4524
```

```
#>
#> Coefficients:
#>                  Estimate Std. Error z value Pr(>|z|)
#> (Intercept)      0.256144   0.137348   1.865 0.062191 .
#> SexWomen        -0.216418   0.072636  -2.979 0.002887 **
#> MarriedMarried   0.150489   0.082097   1.833 0.066791 .
#> Num_Kid5        -0.176415   0.052813  -3.340 0.000837 ***
#> PhD_Quality      0.015271   0.035873   0.426 0.670326
#> Num_MentArticle  0.029082   0.003214   9.048  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for Negative Binomial(2.2644) family taken to be 1)
#>
#>     Null deviance: 1109.0  on 914  degrees of freedom
#> Residual deviance: 1004.3  on 909  degrees of freedom
#> AIC: 3135.9
#>
#> Number of Fisher Scoring iterations: 1
#>
#>
#>               Theta:  2.264
#>           Std. Err.:  0.271
#>
#>  2 x log-likelihood:  -3121.917
```

Interpretation:

- The coefficients are on the log scale.
- The dispersion parameter θ (also called size parameter in some contexts) is estimated as 2.264 with a standard error of 0.271. Check Over-Dispersion for more detail.
- Since θ is significantly different from 1, this confirms overdispersion, validating the choice of the Negative Binomial model over Poisson regression.

7.5.2.2 Model Comparison: Poisson vs. Negative Binomial

7.5.2.2.1 Checking Overdispersion in Poisson Model Before using NB regression, we confirm **overdispersion** by computing:

$$\hat{\phi} = \frac{\text{deviance}}{\text{degrees of freedom}}$$

```r
# Overdispersion check for Poisson model
Poisson_Mod$deviance / Poisson_Mod$df.residual
#> [1] 1.797988
```

- If $\hat{\phi} > 1$, overdispersion is present.
- A large value suggests that **Poisson regression underestimates variance**.

7.5.2.2.2 Likelihood Ratio Test: Poisson vs. Negative Binomial We compare the Poisson and Negative Binomial models using a **likelihood ratio test**, where:

$$G^2 = 2 \times (\log L_{NB} - \log L_{Poisson})$$

with $\text{df} = 1$

```r
# Likelihood ratio test between Poisson and Negative Binomial
pchisq(2 * (logLik(NegBinom_Mod) - logLik(Poisson_Mod)),
       df = 1,
       lower.tail = FALSE)
#> 'log Lik.' 4.391728e-41 (df=7)
```

- **Small p-value (< 0.05)** $\rightarrow$ Negative Binomial model is **significantly better**.
- **Large p-value (> 0.05)** $\rightarrow$ Poisson model **is adequate**.

Since **overdispersion is confirmed**, the Negative Binomial model is preferred.

7.5.2.3 Model Diagnostics and Evaluation

7.5.2.3.1 Checking Dispersion Parameter θ The Negative Binomial dispersion parameter θ can be retrieved:

```r
# Extract dispersion parameter estimate
NegBinom_Mod$theta
#> [1] 2.264388
```

- A large θ suggests that overdispersion is **not extreme**, while a small θ (close to 0) would indicate the Poisson model is reasonable.

7.5.2.4 Predictions and Rate Ratios

In Negative Binomial regression, exponentiating the coefficients gives **rate ratios**:

```r
# Convert coefficients to rate ratios
data.frame(`Odds Ratios` = exp(coef(NegBinom_Mod)))
#>                  Odds.Ratios
#> (Intercept)        1.2919388
```

```
#> SexWomen          0.8053982
#> MarriedMarried    1.1624030
#> Num_Kid5          0.8382698
#> PhD_Quality       1.0153884
#> Num_MentArticle   1.0295094
```

A rate ratio of:

- $> \mathbf{1} \rightarrow$ Increases expected article count.
- $< \mathbf{1} \rightarrow$ Decreases expected article count.
- $= \mathbf{1} \rightarrow$ No effect.

For example:

- If `PhD_Quality` has an exponentiated coefficient of **1.5**, individuals from higher-quality PhD programs are expected to publish **50% more articles**.
- If `Sex` has an exponentiated coefficient of **0.8**, females publish **20% fewer articles** than males, all else equal.

7.5.2.5 Alternative Approach: Zero-Inflated Models

If a dataset has **excess zeros** (many individuals publish **no articles**), **Zero-Inflated Negative Binomial (ZINB) models** may be required.

$$\mathrm{P}(Y_i = 0) = p + (1 - p) f(Y_i = 0 | \mu, \theta)$$

where:

- p is the probability of **always being a zero** (e.g., inactive researchers).
- $f(Y_i)$ follows the **Negative Binomial distribution**.

7.5.3 Fitting a Zero-Inflated Negative Binomial Model

```r
# Load package for zero-inflated models
library(pscl)

# Fit ZINB model
ZINB_Mod <- zeroinfl(Num_Article ~ ., data = bioChemists, dist = "negbin")

# Model summary
summary(ZINB_Mod)
#>
#> Call:
#> zeroinfl(formula = Num_Article ~ ., data = bioChemists, dist = "negbin")
#>
```

```
#> Pearson residuals:
#>     Min      1Q  Median      3Q     Max
#> -1.2942 -0.7601 -0.2909  0.4448  6.4155
#>
#> Count model coefficients (negbin with log link):
#>                   Estimate Std. Error z value Pr(>|z|)
#> (Intercept)      0.4167466  0.1435964   2.902  0.00371 **
#> SexWomen        -0.1955076  0.0755926  -2.586  0.00970 **
#> MarriedMarried   0.0975826  0.0844520   1.155  0.24789
#> Num_Kid5        -0.1517321  0.0542061  -2.799  0.00512 **
#> PhD_Quality     -0.0006998  0.0362697  -0.019  0.98461
#> Num_MentArticle  0.0247862  0.0034927   7.097 1.28e-12 ***
#> Log(theta)       0.9763577  0.1354696   7.207 5.71e-13 ***
#>
#> Zero-inflation model coefficients (binomial with logit link):
#>                 Estimate Std. Error z value Pr(>|z|)
#> (Intercept)     -0.19161    1.32280  -0.145  0.88483
#> SexWomen         0.63587    0.84890   0.749  0.45382
#> MarriedMarried  -1.49944    0.93866  -1.597  0.11017
#> Num_Kid5         0.62841    0.44277   1.419  0.15583
#> PhD_Quality     -0.03773    0.30801  -0.123  0.90250
#> Num_MentArticle -0.88227    0.31622  -2.790  0.00527 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Theta = 2.6548
#> Number of iterations in BFGS optimization: 27
#> Log-likelihood: -1550 on 13 Df
```

This model accounts for:

- Structural **zero inflation**.
- Overdispersion.

ZINB is often preferred when many observations are zero. However, since ZINB does not fall under the GLM framework, we will discuss it further in Nonlinear and Generalized Linear Mixed Models.

Why ZINB is Not a GLM?

- Unlike GLMs, which assume a single response distribution from the exponential family, ZINB is a mixture model with two components:
  - Count model – A negative binomial regression for the main count process.
  - Inflation model – A logistic regression for excess zeros.

Because ZINB combines two distinct processes rather than using a single exponential family distribution, it does not fit within the standard GLM framework.

What ZINB Belongs To

ZINB is part of finite mixture models and is sometimes considered within generalized linear mixed models (GLMMs) or semi-parametric models.

7.6 Quasi-Poisson Regression

Poisson regression assumes that the **mean and variance are equal**:

$$\text{Var}(Y_i) = E(Y_i) = \mu_i$$

However, many real-world datasets exhibit **overdispersion**, where the variance exceeds the mean:

$$\text{Var}(Y_i) = \phi\mu_i$$

where ϕ (the **dispersion parameter**) allows the variance to scale beyond the Poisson assumption.

To correct for this, we use Quasi-Poisson regression, which:

- Follows the Generalized Linear Models structure but is not a strict GLM.
- Uses a variance function proportional to the mean: $\text{Var}(Y_i) = \phi\mu_i$.
- Does not assume a specific probability distribution, unlike Poisson or Negative Binomial models.

7.6.1 Is Quasi-Poisson Regression a Generalized Linear Model?

Yes, Quasi-Poisson is GLM-like:

- **Linear Predictor:** Like Poisson regression, it models the log of the expected count as a function of predictors:

$$\log(E(Y)) = X\beta$$

- **Canonical Link Function:** It typically uses a log link function, just like standard Poisson regression.
- **Variance Structure:** Unlike standard Poisson, which assumes $\text{Var}(Y) = E(Y)$, Quasi-Poisson allows for **overdispersion**:

$$\text{Var}(Y) = \phi E(Y)$$

  where ϕ is estimated rather than assumed to be 1.

No, Quasi-Poisson is not a strict GLM because:

- GLMs require a full probability distribution from the exponential family.
  - Standard Poisson regression assumes a Poisson distribution (which belongs to the exponential family).
  - Quasi-Poisson does not assume a full probability distribution, only a mean-variance relationship.
- It does not use Maximum Likelihood Estimation.
  - Standard GLMs use MLE to estimate parameters.
  - Quasi-Poisson uses quasi-likelihood methods, which require specifying only the mean and variance, but not a full likelihood function.
- Likelihood-based inference is not valid.
  - AIC, BIC, and Likelihood Ratio Tests cannot be used with Quasi-Poisson regression.

When to Use Quasi-Poisson:

- When **data exhibit overdispersion** (variance > mean), making standard Poisson regression inappropriate.
- When Negative Binomial Regression is not preferred, but an alternative is needed to handle overdispersion.
- If overdispersion is present, Negative Binomial Regression is often a better alternative because it is a true GLM with a full likelihood function, whereas Quasi-Poisson is only a quasi-likelihood approach.

7.6.2 Application: Quasi-Poisson Regression

We analyze the `bioChemists` dataset, modeling the number of published articles (`Num_Article`) as a function of various predictors.

7.6.2.1 Checking Overdispersion in the Poisson Model

We first fit a Poisson regression model and check for overdispersion using the deviance-to-degrees-of-freedom ratio:

```
# Fit Poisson regression model
Poisson_Mod <-
    glm(Num_Article ~ ., family = poisson, data = bioChemists)

# Compute dispersion parameter
dispersion_estimate <-
    Poisson_Mod$deviance / Poisson_Mod$df.residual
dispersion_estimate
#> [1] 1.797988
```

- If $\hat{\phi} > 1$, the Poisson model underestimates variance.
- A **large value (» 1)** suggests that Poisson regression **is not appropriate**.

7.6.2.2 Fitting the Quasi-Poisson Model

Since overdispersion is present, we refit the model using Quasi-Poisson regression, which scales standard errors by ϕ.

```
# Fit Quasi-Poisson regression model
quasiPoisson_Mod <-
    glm(Num_Article ~ ., family = quasipoisson, data = bioChemists)

# Summary of the model
summary(quasiPoisson_Mod)
#>
#> Call:
#> glm(formula = Num_Article ~ ., family = quasipoisson, data = bioChemists)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -3.5672  -1.5398  -0.3660   0.5722   5.4467
#>
#> Coefficients:
#>                  Estimate Std. Error t value Pr(>|t|)
#> (Intercept)      0.304617   0.139273   2.187 0.028983 *
#> SexWomen        -0.224594   0.073860  -3.041 0.002427 **
#> MarriedMarried   0.155243   0.083003   1.870 0.061759 .
#> Num_Kid5        -0.184883   0.054268  -3.407 0.000686 ***
#> PhD_Quality      0.012823   0.035700   0.359 0.719544
#> Num_MentArticle  0.025543   0.002713   9.415  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for quasipoisson family taken to be 1.829006)
#>
#>     Null deviance: 1817.4  on 914  degrees of freedom
#> Residual deviance: 1634.4  on 909  degrees of freedom
#> AIC: NA
#>
#> Number of Fisher Scoring iterations: 5
```

Interpretation:

- The **coefficients remain the same** as in Poisson regression.

- **Standard errors are inflated** to account for overdispersion.
- **P-values increase**, leading to more conservative inference.

7.6.2.3 Comparing Poisson and Quasi-Poisson

To see the effect of using **Quasi-Poisson**, we compare **standard errors**:

```
# Extract coefficients and standard errors
poisson_se <- summary(Poisson_Mod)$coefficients[, 2]
quasi_se <- summary(quasiPoisson_Mod)$coefficients[, 2]

# Compare standard errors
se_comparison <- data.frame(Poisson = poisson_se,
                            Quasi_Poisson = quasi_se)
se_comparison
#>                     Poisson Quasi_Poisson
#> (Intercept)     0.102981443   0.139272885
#> SexWomen        0.054613488   0.073859696
#> MarriedMarried  0.061374395   0.083003199
#> Num_Kid5        0.040126898   0.054267922
#> PhD_Quality     0.026397045   0.035699564
#> Num_MentArticle 0.002006073   0.002713028
```

- **Quasi-Poisson has larger standard errors** than Poisson.
- This leads to **wider confidence intervals**, reducing the likelihood of false positives.

7.6.2.4 Model Diagnostics: Checking Residuals

We examine **residuals** to assess model fit:

```
# Residual plot
plot(
    quasiPoisson_Mod$fitted.values,
    residuals(quasiPoisson_Mod, type = "pearson"),
    xlab = "Fitted Values",
    ylab = "Pearson Residuals",
    main = "Residuals vs. Fitted Values (Quasi-Poisson)"
)
abline(h = 0, col = "red")
```

- If residuals show a **pattern**, additional predictors or transformations may be needed.
- **Random scatter around zero** suggests a well-fitting model.

7.6.2.5 Alternative: Negative Binomial vs. Quasi-Poisson

If overdispersion is **severe**, Negative Binomial regression may be preferable because it explicitly models dispersion:

```
# Fit Negative Binomial model
library(MASS)
NegBinom_Mod <- glm.nb(Num_Article ~ ., data = bioChemists)

# Model summaries
summary(quasiPoisson_Mod)
#>
#> Call:
#> glm(formula = Num_Article ~ ., family = quasipoisson, data = bioChemists)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -3.5672  -1.5398  -0.3660   0.5722   5.4467
#>
#> Coefficients:
#>                  Estimate Std. Error t value Pr(>|t|)
#> (Intercept)      0.304617   0.139273   2.187 0.028983 *
#> SexWomen        -0.224594   0.073860  -3.041 0.002427 **
```

```
#> MarriedMarried   0.155243   0.083003   1.870 0.061759 .
#> Num_Kid5        -0.184883   0.054268  -3.407 0.000686 ***
#> PhD_Quality      0.012823   0.035700   0.359 0.719544
#> Num_MentArticle  0.025543   0.002713   9.415  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for quasipoisson family taken to be 1.829006)
#>
#>     Null deviance: 1817.4  on 914  degrees of freedom
#> Residual deviance: 1634.4  on 909  degrees of freedom
#> AIC: NA
#>
#> Number of Fisher Scoring iterations: 5
summary(NegBinom_Mod)
#>
#> Call:
#> glm.nb(formula = Num_Article ~ ., data = bioChemists, init.theta = 2.264387695,
#>     link = log)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -2.1678  -1.3617  -0.2806   0.4476   3.4524
#>
#> Coefficients:
#>                  Estimate Std. Error z value Pr(>|z|)
#> (Intercept)      0.256144   0.137348   1.865 0.062191 .
#> SexWomen        -0.216418   0.072636  -2.979 0.002887 **
#> MarriedMarried   0.150489   0.082097   1.833 0.066791 .
#> Num_Kid5        -0.176415   0.052813  -3.340 0.000837 ***
#> PhD_Quality      0.015271   0.035873   0.426 0.670326
#> Num_MentArticle  0.029082   0.003214   9.048  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for Negative Binomial(2.2644) family taken to be 1)
#>
#>     Null deviance: 1109.0  on 914  degrees of freedom
#> Residual deviance: 1004.3  on 909  degrees of freedom
#> AIC: 3135.9
#>
#> Number of Fisher Scoring iterations: 1
#>
#>
#>               Theta:  2.264
```

```
#>           Std. Err.:  0.271
#>
#>  2 x log-likelihood:  -3121.917
```

7.6.2.6 Key Differences: Quasi-Poisson vs. Negative Binomial

Feature	Quasi-Poisson	Negative Binomial
Handles Overdispersion?	Yes	Yes
Uses a Full Probability Distribution?	No	Yes
MLE-Based?	No (quasi-likelihood)	Yes
Can Use AIC/BIC for Model Selection?	No	Yes
Better for Model Interpretation?	Yes	Yes
Best for Severe Overdispersion?	No	Yes

When to Choose:

- Use Quasi-Poisson when you **only need robust standard errors** and do not require model selection via AIC/BIC.
- Use Negative Binomial when overdispersion is **large** and you want a **true likelihood-based model**.

While Quasi-Poisson is a quick fix, Negative Binomial is generally the better choice for modeling count data with overdispersion.

7.7 Multinomial Logistic Regression

When dealing with categorical response variables with more than two possible outcomes, the **multinomial logistic regression** is a natural extension of the binary logistic model.

7.7.1 The Multinomial Distribution

Suppose we have a categorical response variable Y_i that can take values in $\{1, 2, \dots, J\}$. For each observation i, the probability that it falls into category j is given by:

$$p_{ij} = P(Y_i = j), \quad \text{where} \quad \sum_{j=1}^{J} p_{ij} = 1.$$

The response follows a **multinomial distribution**:

$$Y_i \sim \text{Multinomial}(1; p_{i1}, p_{i2}, ..., p_{iJ}).$$

This means that each observation belongs to exactly one of the J categories.

7.7.2 Modeling Probabilities Using Log-Odds

We cannot model the probabilities p_{ij} directly because they must sum to 1. Instead, we use a **logit transformation**, comparing each category j to a **baseline category** (typically the first category, $j = 1$):

$$\eta_{ij} = \log \frac{p_{ij}}{p_{i1}}, \quad j = 2, \dots, J.$$

Using a **linear function of covariates** $\mathbf{x}_i$, we define:

$$\eta_{ij} = \mathbf{x}_i' \beta_j = \beta_{j0} + \sum_{p=1}^{P} \beta_{jp} x_{ip}.$$

Rearranging to express probabilities explicitly:

$$p_{ij} = p_{i1} \exp(\mathbf{x}_i' \beta_j).$$

Since all probabilities must sum to 1:

$$p_{i1} + \sum_{j=2}^{J} p_{ij} = 1.$$

Substituting for p_{ij}:

$$p_{i1} + \sum_{j=2}^{J} p_{i1} \exp(\mathbf{x}_i' \beta_j) = 1.$$

Solving for p_{i1}:

$$p_{i1} = \frac{1}{1 + \sum_{j=2}^{J} \exp(\mathbf{x}_i' \beta_j)}.$$

Thus, the probability for category j is:

$$p_{ij} = \frac{\exp(\mathbf{x}_i'\beta_j)}{1 + \sum_{l=2}^{J} \exp(\mathbf{x}_i'\beta_l)}, \quad j = 2, \dots, J.$$

This formulation is known as the **multinomial logit model**.

7.7.3 Softmax Representation

An alternative formulation avoids choosing a baseline category and instead treats all J categories **symmetrically** using the **softmax function**:

$$P(Y_i = j | X_i = x) = \frac{\exp(\beta_{j0} + \sum_{p=1}^{P} \beta_{jp} x_p)}{\sum_{l=1}^{J} \exp(\beta_{l0} + \sum_{p=1}^{P} \beta_{lp} x_p)}.$$

This representation is often used in **neural networks** and general machine learning models.

7.7.4 Log-Odds Ratio Between Two Categories

The **log-odds ratio** between two categories k and k' is:

$$\log \frac{P(Y = k | X = x)}{P(Y = k' | X = x)} = (\beta_{k0} - \beta_{k'0}) + \sum_{p=1}^{P} (\beta_{kp} - \beta_{k'p}) x_p.$$

This equation tells us that:

- If $\beta_{kp} > \beta_{k'p}$, then increasing x_p increases the odds of choosing category k over k'.
- If $\beta_{kp} < \beta_{k'p}$, then increasing x_p decreases the odds of choosing k over k'.

7.7.5 Estimation

To estimate the parameters β_j, we use Maximum Likelihood estimation.

Given n independent observations (Y_i, X_i), the likelihood function is:

$$L(\beta) = \prod_{i=1}^{n} \prod_{j=1}^{J} p_{ij}^{Y_{ij}}.$$

Taking the log-likelihood:

$$\log L(\beta) = \sum_{i=1}^{n} \sum_{j=1}^{J} Y_{ij} \log p_{ij}.$$

Since there is no closed-form solution, numerical methods (see Non-linear Least Squares Estimation) are used for estimation.

7.7.6 Interpretation of Coefficients

- Each β_{jp} represents the effect of x_p on the **log-odds** of category j relative to the baseline.
- **Positive coefficients** mean increasing x_p makes category j more likely relative to the baseline.
- **Negative coefficients** mean increasing x_p makes category j less likely relative to the baseline.

7.7.7 Application: Multinomial Logistic Regression

1. Load Necessary Libraries and Data

```
library(faraway)  # For the dataset
library(dplyr)    # For data manipulation
library(ggplot2)  # For visualization
library(nnet)     # For multinomial logistic regression

# Load and inspect data
data(nes96, package="faraway")
head(nes96, 3)
#>   popul TVnews selfLR ClinLR DoleLR     PID age  educ   income    vote
#> 1     0      7 extCon extLib    Con  strRep  36    HS $3Kminus    Dole
#> 2   190      1 sliLib sliLib sliCon weakDem  20  Coll $3Kminus Clinton
#> 3    31      7    Lib    Lib    Con weakDem  24 BAdeg $3Kminus Clinton
```

The dataset `nes96` contains survey responses, including political party identification (`PID`), age (`age`), and education level (`educ`).

2. Define Political Strength Categories

We classify political strength into three categories:

1. **Strong**: Strong Democrat or Strong Republican
2. **Weak**: Weak Democrat or Weak Republican
3. **Neutral**: Independents and other affiliations

```
# Check distribution of political identity
table(nes96$PID)
#>
```

```
#>  strDem weakDem  indDem  indind  indRep weakRep  strRep
#>     200     180     108      37      94     150     175

# Define Political Strength variable
nes96 <- nes96 %>%
  mutate(Political_Strength = case_when(
    PID %in% c("strDem", "strRep") ~ "Strong",
    PID %in% c("weakDem", "weakRep") ~ "Weak",
    PID %in% c("indDem", "indind", "indRep") ~ "Neutral",
    TRUE ~ NA_character_
  ))

# Summarize
nes96 %>% group_by(Political_Strength) %>% summarise(Count = n())
#> # A tibble: 3 x 2
#>   Political_Strength Count
#>   <chr>              <int>
#> 1 Neutral              239
#> 2 Strong               375
#> 3 Weak                 330
```

3. Visualizing Political Strength by Age

We visualize the proportion of each political strength category across age groups.

```
# Prepare data for visualization
Plot_DF <- nes96 %>%
    mutate(Age_Grp = cut_number(age, 4)) %>%
    group_by(Age_Grp, Political_Strength) %>%
    summarise(count = n(), .groups = 'drop') %>%
    group_by(Age_Grp) %>%
    mutate(etotal = sum(count), proportion = count / etotal)

# Plot age vs political strength
Age_Plot <- ggplot(
    Plot_DF,
    aes(
        x        = Age_Grp,
        y        = proportion,
        group    = Political_Strength,
        linetype = Political_Strength,
        color    = Political_Strength
    )
) +
    geom_line(size = 2) +
    labs(title = "Political Strength by Age Group",
```

```
         x = "Age Group",
         y = "Proportion")

# Display plot
Age_Plot
```

4. Fit a Multinomial Logistic Model

We model **political strength** as a function of **age** and **education**.

```
# Fit multinomial logistic regression
Multinomial_Model <-
    multinom(Political_Strength ~ age + educ,
             data = nes96,
             trace = FALSE)
summary(Multinomial_Model)
#> Call:
#> multinom(formula = Political_Strength ~ age + educ, data = nes96,
#>     trace = FALSE)
#>
#> Coefficients:
#>        (Intercept)          age     educ.L     educ.Q     educ.C      educ^4
#> Strong -0.08788729  0.010700364 -0.1098951 -0.2016197 -0.1757739 -0.02116307
#> Weak    0.51976285 -0.004868771 -0.1431104 -0.2405395 -0.2411795  0.18353634
#>            educ^5     educ^6
#> Strong -0.1664377 -0.1359449
#> Weak   -0.1489030 -0.2173144
#>
```

```
#> Std. Errors:
#>        (Intercept)         age    educ.L    educ.Q    educ.C    educ^4
#> Strong   0.3017034 0.005280743 0.4586041 0.4318830 0.3628837 0.2964776
#> Weak     0.3097923 0.005537561 0.4920736 0.4616446 0.3881003 0.3169149
#>           educ^5    educ^6
#> Strong 0.2515012 0.2166774
#> Weak   0.2643747 0.2199186
#>
#> Residual Deviance: 2024.596
#> AIC: 2056.596
```

5. Stepwise Model Selection Based on AIC

We perform stepwise selection to find the best model.

```
Multinomial_Step <- step(Multinomial_Model, trace = 0)
#> trying - age
#> trying - educ
#> trying - age
Multinomial_Step
#> Call:
#> multinom(formula = Political_Strength ~ age, data = nes96, trace = FALSE)
#>
#> Coefficients:
#>        (Intercept)          age
#> Strong -0.01988977  0.009832916
#> Weak    0.59497046 -0.005954348
#>
#> Residual Deviance: 2030.756
#> AIC: 2038.756
```

Compare the best model to the full model based on **deviance**:

```
pchisq(
    q = deviance(Multinomial_Step) - deviance(Multinomial_Model),
    df = Multinomial_Model$edf - Multinomial_Step$edf,
    lower.tail = FALSE
)
#> [1] 0.9078172
```

A non-significant p-value suggests **no major difference** between the full and stepwise models.

6. Predictions & Visualization

Predicting Political Strength Probabilities by Age

```
# Create data for prediction
PlotData <- data.frame(age = seq(from = 19, to = 91))
```

```r
# Get predicted probabilities
Preds <- PlotData %>%
    bind_cols(data.frame(predict(Multinomial_Step,
                                 PlotData,
                                 type = "probs")))

# Plot predicted probabilities across age
plot(
    x = Preds$age,
    y = Preds$Neutral,
    type = "l",
    ylim = c(0.2, 0.6),
    col = "black",
    ylab = "Proportion",
    xlab = "Age"
)

lines(x = Preds$age,
      y = Preds$Weak,
      col = "blue")
lines(x = Preds$age,
      y = Preds$Strong,
      col = "red")

legend(
    "topleft",
    legend = c("Neutral", "Weak", "Strong"),
    col = c("black", "blue", "red"),
    lty = 1
)
```

Predict for Specific Ages

```
# Predict class for a 34-year-old
predict(Multinomial_Step, data.frame(age = 34))
#> [1] Weak
#> Levels: Neutral Strong Weak

# Predict probabilities for 34 and 35-year-olds
predict(Multinomial_Step, data.frame(age = c(34, 35)), type = "probs")
#>     Neutral    Strong      Weak
#> 1 0.2597275 0.3556910 0.3845815
#> 2 0.2594080 0.3587639 0.3818281
```

7.7.8 Application: Gamma Regression

When response variables are **strictly positive**, we use **Gamma regression**.

1. Load and Prepare Data

```
library(agridat)  # Agricultural dataset

# Load and filter data
dat <- agridat::streibig.competition
gammaDat <- subset(dat, sseeds < 1)  # Keep only barley
gammaDat <-
    transform(gammaDat,
              x = bseeds,
              y = bdwt,
```

```
                 block = factor(block))
```

2. Visualization of Inverse Yield

```
ggplot(gammaDat, aes(x = x, y = 1 / y)) +
    geom_point(aes(color = block, shape = block)) +
    labs(title = "Inverse Yield vs Seeding Rate",
         x = "Seeding Rate",
         y = "Inverse Yield")
```

3. Fit Gamma Regression Model

Gamma regression models **yield as a function of seeding rate** using an inverse link:

$$\eta_{ij} = \beta_{0j} + \beta_{1j}x_{ij} + \beta_2 x_{ij}^2, \quad Y_{ij} = \eta_{ij}^{-1}$$

```
m1 <- glm(y ~ block + block * x + block * I(x^2),
          data = gammaDat, family = Gamma(link = "inverse"))

summary(m1)
#>
#> Call:
#> glm(formula = y ~ block + block * x + block * I(x^2), family = Gamma(link = "inverse"),
#>     data = gammaDat)
#>
#> Deviance Residuals:
#>      Min        1Q    Median        3Q       Max
#> -1.21708  -0.44148   0.02479   0.17999   0.80745
```

```
#>
#> Coefficients:
#>                   Estimate Std. Error t value Pr(>|t|)
#> (Intercept)     1.115e-01  2.870e-02   3.886 0.000854 ***
#> blockB2        -1.208e-02  3.880e-02  -0.311 0.758630
#> blockB3        -2.386e-02  3.683e-02  -0.648 0.524029
#> x              -2.075e-03  1.099e-03  -1.888 0.072884 .
#> I(x^2)          1.372e-05  9.109e-06   1.506 0.146849
#> blockB2:x       5.198e-04  1.468e-03   0.354 0.726814
#> blockB3:x       7.475e-04  1.393e-03   0.537 0.597103
#> blockB2:I(x^2) -5.076e-06  1.184e-05  -0.429 0.672475
#> blockB3:I(x^2) -6.651e-06  1.123e-05  -0.592 0.560012
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for Gamma family taken to be 0.3232083)
#>
#>     Null deviance: 13.1677  on 29  degrees of freedom
#> Residual deviance:  7.8605  on 21  degrees of freedom
#> AIC: 225.32
#>
#> Number of Fisher Scoring iterations: 5
```

4. Predictions and Visualization

```r
# Generate new data for prediction
newdf <-
    expand.grid(x = seq(0, 120, length = 50),
                block = factor(c("B1", "B2", "B3")))

# Predict responses
newdf$pred <- predict(m1, newdata = newdf, type = "response")

# Plot predictions
ggplot(gammaDat, aes(x = x, y = y)) +
    geom_point(aes(color = block, shape = block)) +
    geom_line(data = newdf, aes(
        x = x,
        y = pred,
        color = block,
        linetype = block
    )) +
    labs(title = "Predicted Yield by Seeding Rate",
         x = "Seeding Rate",
         y = "Yield")
```

7.8 Generalization of Generalized Linear Models

We have seen that Poisson regression bears similarities to logistic regression. This insight leads us to a broader class of models known as Generalized Linear Models, introduced by Nelder and Wedderburn (1972). These models provide a unified framework for handling different types of response variables while maintaining the fundamental principles of linear modeling.

7.8.1 Exponential Family

The foundation of GLMs is built on the **exponential family of distributions**, which provides a flexible class of probability distributions that share a common form:

$$f(y;\theta,\phi) = \exp\left(\frac{\theta y - b(\theta)}{a(\phi)} + c(y,\phi)\right)$$

where:

- θ is the **natural parameter** (canonical parameter),
- ϕ is the **dispersion parameter**,
- $a(\phi)$, $b(\theta)$, and $c(y,\phi)$ are functions ensuring the proper distributional form.

Distributions in the Exponential Family that Can Be Used in GLMs:

1. Normal Distribution
2. Binomial Distribution
3. Poisson Distribution
4. Gamma Distribution
5. Inverse Gaussian Distribution
6. Negative Binomial Distribution (used in GLMs but requires overdispersion adjustments)
7. Multinomial Distribution (for categorical response)

Exponential Family Distributions Not Commonly Used in GLMs:

1. Beta Distribution
2. Dirichlet Distribution
3. Wishart Distribution
4. Geometric Distribution
5. Exponential Distribution (can be used indirectly through survival models)

Example: Normal Distribution

Consider a normally distributed response variable $Y \sim N(\mu, \sigma^2)$. The probability density function (PDF) is:

$$\begin{aligned} f(y; \mu, \sigma^2) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right) \\ &= \exp\left(-\frac{y^2 - 2y\mu + \mu^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2)\right) \end{aligned}$$

Rewriting in exponential family form:

$$\begin{aligned} f(y; \mu, \sigma^2) &= \exp\left(\frac{y\mu - \frac{\mu^2}{2}}{\sigma^2} - \frac{y^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2)\right) \\ &= \exp\left(\frac{\theta y - b(\theta)}{a(\phi)} + c(y, \phi)\right) \end{aligned}$$

where:

- **Natural parameter**: $\theta = \mu$
- **Function** $b(\theta)$: $b(\theta) = \frac{\mu^2}{2}$
- **Dispersion function**: $a(\phi) = \sigma^2 = \phi$
- **Function** $c(y, \phi)$: $c(y, \phi) = -\frac{1}{2}\left(\frac{y^2}{\phi} + \log(2\pi\sigma^2)\right)$

7.8.2 Properties of GLM Exponential Families

1. **Expected Value (Mean)**

$$E(Y) = b'(\theta)$$

where $b'(\theta) = \frac{\partial b(\theta)}{\partial \theta}$. (Note: ' is "prime," not transpose).

2. **Variance**

$$\mathrm{Var}(Y) = a(\phi)b''(\theta) = a(\phi)V(\mu)$$

where:

- $V(\mu) = b''(\theta)$ is the **variance function**, though it only represents the variance when $a(\phi) = 1$.

3. If $a(\phi)$, $b(\theta)$, and $c(y, \phi)$ are identifiable, we can derive the expected value and variance of Y.

Examples of Exponential Family Distributions

1. Normal Distribution

For a normal distribution $Y \sim N(\mu, \sigma^2)$, the exponential family representation is:

$$f(y; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$$

which can be rewritten in exponential form:

$$\exp\left(\frac{y\mu - \frac{1}{2}\mu^2}{\sigma^2} - \frac{y^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2)\right)$$

From this, we identify:

- $\theta = \mu$
- $b(\theta) = \frac{\theta^2}{2}$
- $a(\phi) = \sigma^2$

Computing derivatives:

$$b'(\theta) = \frac{\partial b(\theta)}{\partial \theta} = \mu, \quad V(\mu) = b''(\theta) = 1$$

Thus,

$$E(Y) = \mu, \quad \mathrm{Var}(Y) = a(\phi)V(\mu) = \sigma^2$$

---

2. Poisson Distribution

For a Poisson-distributed response $Y \sim \text{Poisson}(\mu)$, the probability mass function is:

$$f(y;\mu) = \frac{\mu^y e^{-\mu}}{y!}$$

Rewriting in exponential form:

$$\exp(y \log \mu - \mu - \log y!)$$

Thus, we identify:

- $\theta = \log \mu$
- $b(\theta) = e^\theta$
- $a(\phi) = 1$
- $c(y, \phi) = \log y!$

Computing derivatives:

$$E(Y) = b'(\theta) = e^\theta = \mu, \quad \text{Var}(Y) = b''(\theta) = \mu$$

Since $\mu = E(Y)$, we confirm the variance function:

$$\text{Var}(Y) = V(\mu) = \mu$$

7.8.3 Structure of a Generalized Linear Model

In a GLM, we model the mean μ through a **link function** that connects it to a linear predictor:

$$g(\mu) = g(b'(\theta)) = \mathbf{x}'$$

Equivalently,

$$\mu = g^{-1}(\mathbf{x}' \ )$$

where:

- $g(\cdot)$ is the **link function**, which ensures a transformation between the expected response $E(Y) = \mu$ and the linear predictor.

- $\eta = \mathbf{x}'$ is called the **linear predictor**.

7.8.4 Components of a GLM

A GLM consists of two main components:

7.8.4.1 Random Component

This describes the **distribution of the response variable** $Y_1, \dots, Y_n$. The response variables are assumed to follow a distribution from the **exponential family**, which can be written as:

$$f(y_i; \theta_i, \phi) = \exp\left(\frac{\theta_i y_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi)\right)$$

where:

- Y_i are **independent** random variables.
- The **canonical parameter** θ_i may differ for each observation.
- The **dispersion parameter** ϕ is assumed to be constant across all i.
- The mean response is given by:

$$\mu_i = E(Y_i)$$

7.8.4.2 Systematic Component

This specifies how the mean response μ is related to the explanatory variables $\mathbf{x}$ through a **linear predictor** η:

- The systematic component consists of:
  - A **link function** $g(\mu)$.
  - A **linear predictor** $\eta = \mathbf{x}'$.
- **Notation**:
  - We assume:

$$g(\mu_i) = \mathbf{x}' \ = \eta_i$$

  - The parameter vector $= (\beta_1, \dots, \beta_p)'$ needs to be estimated.

7.8.5 Canonical Link

In a GLM, a link function $g(\cdot)$ relates the mean μ_i of the response Y_i to the linear predictor η_i via

$$\eta_i = g(\mu_i).$$

A **canonical link** is a special case of $g(\cdot)$ where

$$g(\mu_i) = \eta_i = \theta_i,$$

and θ_i is the **natural parameter** of the exponential family. In other words, the link function directly equates the linear predictor η_i with the distribution's natural parameter θ_i. Hence, $g(\mu)$ is **canonical** if $g(\mu) = \theta$.

Exponential Family Components

- $b(\theta)$: the **cumulant (moment generating) function**, which defines the variance function.
- $g(\mu)$: the **link function**, which must be
  - Monotonically increasing
  - Continuously differentiable
  - Invertible

Figure 7.1: GLM Structure

For an exponential-family distribution, the function $b(\theta)$ is called the *cumulant moment generating function*, and it relates θ to the mean via its derivative:

$$\mu = b'(\theta) \quad \Longleftrightarrow \quad \theta = b'^{-1}(\mu).$$

By defining the link so that $g(\mu) = \theta$, we impose $\eta_i = \theta_i$, which is why $g(\cdot)$ is termed **canonical** in this setting.

When the link is canonical, an equivalent way to express this is

$$\gamma^{-1} \circ g^{-1} = I,$$

indicating that the inverse link $g^{-1}(\cdot)$ directly maps the linear predictor (now the natural parameter θ) back to μ in a way that respects the structure of the exponential family.

Choosing $g(\cdot)$ to be canonical often simplifies mathematical derivations and computations—especially for parameter estimation—because the linear predictor η and the natural parameter θ coincide. Common examples of canonical links include:

- **Identity link** for the normal (Gaussian) distribution
- **Log link** for the Poisson distribution
- **Logit link** for the Bernoulli (binomial) distribution

In each case, setting $\eta = \theta$ streamlines the relationship between the mean and the linear predictor, making the model both elegant and practically convenient.

7.8.6 Inverse Link Functions

The **inverse link function** $g^{-1}(\eta)$ (also called the **mean function**) transforms the linear predictor η (which can take any real value) into a **valid mean response** μ.

Example 1: Normal Distribution (Identity Link)

- Random Component: $Y_i \sim N(\mu_i, \sigma^2)$.
- Mean Response: $\mu_i = \theta_i$.
- Canonical Link Function:

$$g(\mu_i) = \mu_i$$

  (i.e., the identity function).

Example 2: Binomial Distribution (Logit Link)

- Random Component: $Y_i \sim \text{Binomial}(n_i, p_i)$.
- Mean Response:

$$\mu_i = \frac{n_i e^{\theta_i}}{1 + e^{\theta_i}}$$

- Canonical Link Function:

$$g(\mu_i) = \log\left(\frac{\mu_i}{n_i - \mu_i}\right)$$

(Logit link function).

Example 3: Poisson Distribution (Log Link)

- Random Component: $Y_i \sim \text{Poisson}(\mu_i)$.
- Mean Response:

$$\mu_i = e^{\theta_i}$$

- Canonical Link Function:

$$g(\mu_i) = \log(\mu_i)$$

(Log link function).

Example 4: Gamma Distribution (Inverse Link)

- Random Component: $Y_i \sim \text{Gamma}(\alpha, \mu_i)$.
- Mean Response:

$$\mu_i = -\frac{1}{\theta_i}$$

- Canonical Link Function:

$$g(\mu_i) = -\frac{1}{\mu_i}$$

(Inverse link function).

The following table presents common **GLM link functions** and their corresponding **inverse functions**.

Link	$\eta_i = g(\mu_i)$	$\mu_i = g^{-1}(\eta_i)$
Identity	μ_i	η_i
Log	$\log_e \mu_i$	e^{η_i}
Inverse	μ_i^{-1}	η_i^{-1}
Inverse-square	μ_i^{-2}	$\eta_i^{-1/2}$
Square-root	$\sqrt{\mu_i}$	η_i^2
Logit	$\log_e\left(\frac{\mu_i}{1-\mu_i}\right)$	$\frac{1}{1+e^{-\eta_i}}$
Probit	$\Phi^{-1}(\mu_i)$	$\Phi(\eta_i)$

where

- μ_i is the expected value of the response.
- η_i is the linear predictor.
- $\Phi(\cdot)$ represents the CDF of the standard normal distribution.

7.8.7 Estimation of Parameters in GLMs

The GLM framework extends Linear Regression by allowing for response variables that follow **exponential family distributions**.

- Maximum Likelihood Estimation is used to estimate the parameters of the systematic component (β), providing a consistent and efficient approach. The derivation and computation processes are unified, thanks to the exponential form of the model, which simplifies mathematical treatment and implementation.
- However, this unification does not extend to the estimation of the dispersion parameter (ϕ), which requires separate treatment, often involving alternative estimation methods such as moment-based approaches or quasi-likelihood estimation.

In GLMs, the response variable Y_i follows an **exponential family distribution** characterized by the density function:

$$f(y_i; \theta_i, \phi) = \exp\left(\frac{\theta_i y_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi)\right)$$

where:

- θ_i is the **canonical parameter**.
- ϕ is the **dispersion parameter** (which may be known or estimated separately).
- $b(\theta_i)$ determines the mean and variance of Y_i.
- $a(\phi)$ scales the variance.
- $c(y_i, \phi)$ ensures proper normalization.

For this family, we obtain:

- **Mean of Y_i**:
$$E(Y_i) = \mu_i = b'(\theta_i)$$
- **Variance of Y_i**:
$$\mathrm{Var}(Y_i) = b''(\theta_i)a(\phi) = V(\mu_i)a(\phi)$$
where $V(\mu_i)$ is the **variance function**.

- **Systematic component (link function):**

$$g(\mu_i) = \eta_i = \mathbf{x}_i'\beta$$

The function $g(\cdot)$ is the **link function**, which connects the expected response μ_i to the linear predictor $\mathbf{x}_i'\beta$.

For a single observation Y_i, the log-likelihood function is:

$$l_i(\beta, \phi) = \frac{\theta_i y_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi)$$

For n independent observations, the **total log-likelihood** is:

$$l(\beta, \phi) = \sum_{i=1}^{n} l_i(\beta, \phi)$$

Expanding this,

$$l(\beta, \phi) = \sum_{i=1}^{n} \left(\frac{\theta_i y_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi) \right).$$

To estimate β, we maximize this log-likelihood function.

7.8.7.1 Estimation of Systematic Component (β)

To differentiate $l(\beta, \phi)$ with respect to β_j, we apply the **chain rule**:

$$\frac{\partial l_i(\beta, \phi)}{\partial \beta_j} = \underbrace{\frac{\partial l_i(\beta, \phi)}{\partial \theta_i}}_{\text{depends on } (y_i - \mu_i)} \times \underbrace{\frac{\partial \theta_i}{\partial \mu_i}}_{=1/V(\mu_i) \text{ if canonical link}} \times \underbrace{\frac{\partial \mu_i}{\partial \eta_i}}_{\text{depends on the link}} \times \underbrace{\frac{\partial \eta_i}{\partial \beta_j}}_{=x_{ij}}.$$

Let us see why these four pieces appear:

1. $l_i(\beta, \phi)$ depends on θ_i. So we start by computing $\frac{\partial l_i}{\partial \theta_i}$.
2. θ_i (the "natural parameter" in the exponential family) may in turn be a function of μ_i.
   - In **canonical**-link GLMs, we often have $\theta_i = \eta_i$.
   - In more **general**-link GLMs,θ_i is still some function of μ_i.
     Hence we need $\frac{\partial \theta_i}{\partial \mu_i}$.

3. μ_i (the mean) is a function of the linear predictor η_i. Typically, $\eta_i = g(\mu_i)$ implies $\mu_i = g^{-1}(\eta_i)$. So we need $\frac{\partial \mu_i}{\partial \eta_i}$.
4. Finally, $\eta_i = \mathbf{x}_i'\beta$. So the derivative $\frac{\partial \eta_i}{\partial \beta_j}$ is simply x_{ij}, the j-th component of the covariate vector $\mathbf{x}_i$.

Let us look at each factor in turn.

First term:

$$\frac{\partial l_i(\beta, \phi)}{\partial \theta_i}$$

Recall

$$l_i(\theta_i, \phi) = \frac{\theta_i\, y_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi).$$

1. Differentiate $\theta_i y_i - b(\theta_i)$ with respect to θ_i:

$$\frac{\partial}{\partial \theta_i}[\theta_i\, y_i - b(\theta_i)] = y_i - b'(\theta_i).$$

   But by exponential-family definitions,$b'(\theta_i) = \mu_i$.
   So that is $y_i - \mu_i$.
2. Since everything is divided by $a(\phi)$, we get

$$\frac{\partial l_i}{\partial \theta_i} = \frac{1}{a(\phi)}\,[\,y_i - \mu_i\,].$$

   Hence,

$$\boxed{\frac{\partial l_i(\beta, \phi)}{\partial \theta_i} = \frac{y_i - \mu_i}{a(\phi)}.}$$

Second term:

$$\frac{\partial \theta_i}{\partial \mu_i}$$

1. In an exponential family with **canonical link**, we have

$$\theta_i = \eta_i = \mathbf{x}_i'\beta.$$

Then θ_i is actually the same as η_i, which is the same as $g(\mu_i)$. Recall also that if $\mu_i = b'(\theta_i)$, then $d\mu_i/d\theta_i = b''(\theta_i)$. But $b''(\theta_i) = V(\mu_i)$. Hence

$$\frac{d\mu_i}{d\theta_i} = V(\mu_i) \quad \Longrightarrow \quad \frac{d\theta_i}{d\mu_i} = \frac{1}{V(\mu_i)}.$$

This identity is a well-known property of **canonical** links in the exponential family.

2. In more general (non-canonical) links, θ_i may be some other smooth function of μ_i. The key idea is: if $\mu_i = b'(\theta_i)$ but $\eta_i \neq \theta_i$, you would have to carefully derive $\partial\theta_i/\partial\mu_i$. Often, a canonical link is assumed to keep expressions simpler.

If we **assume** a canonical link throughout, then

$$\boxed{\frac{\partial\theta_i}{\partial\mu_i} = \frac{1}{V(\mu_i)}.}$$

Third term:

$$\frac{\partial\mu_i}{\partial\eta_i}$$

Here we consider the link function $g(\cdot)$, defined by

$$\eta_i = g(\mu_i) \quad \Longrightarrow \quad \mu_i = g^{-1}(\eta_i).$$

For example,

- In a Bernoulli (logistic-regression) model, $\mu g(\mu) = \log\frac{\mu}{1-\mu}$. So $\mu = g^{-1}(\eta) = \frac{1}{1+e^{-\eta}}$. Then $\frac{d\mu}{d\eta} = \mu\,(1-\mu)$.
- For a Poisson (log) link, $g(\mu) = \log(\mu)$. So $\mu = e^{\eta}$. Then $\frac{d\mu}{d\eta} = e^{\eta} = \mu$.
- For an identity link, $g(\mu) = \mu$. Then $\eta = \mu$ and $\frac{d\mu}{d\eta} = 1$.

In general,

$$\boxed{\frac{\partial\mu_i}{\partial\eta_i} = \frac{d}{d\eta}[g^{-1}(\eta)]\Big|_{\eta=\eta_i} = \left(g^{-1}\right)'(\eta_i).}$$

If the link is *also* canonical, then typically $\frac{\partial\mu_i}{\partial\eta_i} = V(\mu_i)$. Indeed, that is consistent with the second term result.

Fourth term:

$$\frac{\partial\eta_i}{\partial\beta_j}$$

Finally, the linear predictor is

$$\eta_i = \mathbf{x}_i'\beta = \sum_{k=1}^{p} x_{ik}\,\beta_k.$$

Hence, the derivative of η_i with respect to β_j is simply

$$\boxed{\frac{\partial \eta_i}{\partial \beta_j} = x_{ij}.}$$

Therefore, for the **entire** log-likelihood $l(\beta, \phi) = \sum_{i=1}^{n} l_i(\beta, \phi)$, we sum over i:

$$\boxed{\frac{\partial l(\beta, \phi)}{\partial \beta_j} = \sum_{i=1}^{n} \Big[\frac{y_i - \mu_i}{a(\phi)} \times \frac{1}{V(\mu_i)} \times \frac{\partial \mu_i}{\partial \eta_i} \times x_{ij} \Big].}$$

To simplify expressions, we define the **weight**:

$$w_i = \left(\left(\frac{\partial \eta_i}{\partial \mu_i} \right)^2 V(\mu_i) \right)^{-1}.$$

For **canonical links**, this often simplifies to $w_i = V(\mu_i)$, such as:

- **Bernoulli (logit link)**: $w_i = p_i(1 - p_i)$.
- **Poisson (log link)**: $w_i = \mu_i$.

Rewriting the score function in terms of w_i:

$$\frac{\partial l(\beta, \phi)}{\partial \beta_j} = \sum_{i=1}^{n} \left[\frac{y_i - \mu_i}{a(\phi)} \times w_i \times \frac{\partial \eta_i}{\partial \mu_i} \times x_{ij} \right].$$

To use the Newton-Raphson Algorithm for estimating β, we require the expected second derivative:

$$-E\left(\frac{\partial^2 l(\beta, \phi)}{\partial \beta_j \partial \beta_k} \right),$$

which corresponds to the (j, k)-th element of the **Fisher information matrix** $\mathbf{I}(\beta)$:

$$\mathbf{I}_{jk}(\beta) = -E\left(\frac{\partial^2 l(\beta, \phi)}{\partial \beta_j \partial \beta_k} \right) = \sum_{i=1}^{n} \frac{w_i}{a(\phi)} x_{ij} x_{ik}.$$

Example: Bernoulli Model with Logit Link

For a **Bernoulli** response with a **logit link function** (canonical link), we have:

$$\begin{aligned} b(\theta) &= \log(1+\exp(\theta)) = \log(1+\exp(\mathbf{x}'\beta)), \\ a(\phi) &= 1, \\ c(y_i,\phi) &= 0. \end{aligned}$$

From the mean and link function:

$$\begin{aligned} E(Y) = b'(\theta) &= \frac{\exp(\theta)}{1+\exp(\theta)} = \mu = p, \\ \eta = g(\mu) &= \log\left(\frac{\mu}{1-\mu}\right) = \theta = \log\left(\frac{p}{1-p}\right) = \mathbf{x}'\beta. \end{aligned}$$

The log-likelihood for Y_i is:

$$l_i(\beta,\phi) = \frac{y_i\theta_i - b(\theta_i)}{a(\phi)} + c(y_i,\phi) = y_i\mathbf{x}'_i\beta - \log(1+\exp(\mathbf{x}'\beta)).$$

We also obtain:

$$\begin{aligned} V(\mu_i) &= \mu_i(1-\mu_i) = p_i(1-p_i), \\ \frac{\partial\mu_i}{\partial\eta_i} &= p_i(1-p_i). \end{aligned}$$

Thus, the first derivative simplifies as:

$$\begin{aligned} \frac{\partial l(\beta,\phi)}{\partial\beta_j} &= \sum_{i=1}^n \left[\frac{y_i-\mu_i}{a(\phi)} \times \frac{1}{V(\mu_i)} \times \frac{\partial\mu_i}{\partial\eta_i} \times x_{ij}\right] \\ &= \sum_{i=1}^n (y_i-p_i) \times \frac{1}{p_i(1-p_i)} \times p_i(1-p_i) \times x_{ij} \\ &= \sum_{i=1}^n (y_i-p_i)x_{ij} \\ &= \sum_{i=1}^n \left(y_i - \frac{\exp(\mathbf{x}'_i\beta)}{1+\exp(\mathbf{x}'_i\beta)}\right) x_{ij}. \end{aligned}$$

The weight term in this case is:

$$w_i = \left(\left(\frac{\partial\eta_i}{\partial\mu_i}\right)^2 V(\mu_i)\right)^{-1} = p_i(1-p_i).$$

Thus, the Fisher information matrix has elements:

$$\mathbf{I}_{jk}(\beta) = \sum_{i=1}^{n} \frac{w_i}{a(\phi)} x_{ij} x_{ik} = \sum_{i=1}^{n} p_i(1-p_i) x_{ij} x_{ik}.$$

‘

The **Fisher-Scoring** algorithm for the Maximum Likelihood estimate of is given by:

$$\begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}^{(m+1)} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}^{(m)} + \mathbf{I}^{-1}(\ ) \begin{pmatrix} \frac{\partial l(\beta,\phi)}{\partial \beta_1} \\ \frac{\partial l(\beta,\phi)}{\partial \beta_2} \\ \vdots \\ \frac{\partial l(\beta,\phi)}{\partial \beta_p} \end{pmatrix} \Bigg|_{\beta=\beta^{(m)}}$$

This is similar to the Newton-Raphson Algorithm, except that we replace the observed matrix of second derivatives (Hessian) with its expected value.

In matrix representation, the score function (gradient of the log-likelihood) is:

$$\begin{aligned} \frac{\partial l}{\partial \beta} &= \frac{1}{a(\phi)} \mathbf{X}'\mathbf{W}\ (\mathbf{y} - \ ) \\ &= \frac{1}{a(\phi)} \mathbf{F}'\mathbf{V}^{-1}(\mathbf{y} - \ ) \end{aligned}$$

The expected Fisher information matrix is:

$$\mathbf{I}(\beta) = \frac{1}{a(\phi)} \mathbf{X}'\mathbf{W}\mathbf{X} = \frac{1}{a(\phi)} \mathbf{F}'\mathbf{V}^{-1}\mathbf{F}$$

where:

- $\mathbf{X}$ is an $n \times p$ matrix of covariates.
- $\mathbf{W}$ is an $n \times n$ diagonal matrix with (i,i)-th element given by w_i.
- is an $n \times n$ diagonal matrix with (i,i)-th element given by $\frac{\partial \eta_i}{\partial \mu_i}$.
- $\mathbf{F} = \frac{\partial \mu}{\partial \beta}$ is an $n \times p$ matrix, where the i-th row is given by $\frac{\partial \mu_i}{\partial \beta} = (\frac{\partial \mu_i}{\partial \eta_i})\mathbf{x}_i'$.
- $\mathbf{V}$ is an $n \times n$ diagonal matrix with (i,i)-th element given by $V(\mu_i)$.

Setting the derivative of the log-likelihood equal to zero gives the MLE equations:

$$\mathbf{F}'\mathbf{V}^{-1}\mathbf{y} = \mathbf{F}'\mathbf{V}^{-1}$$

Since all components of this equation (except for $\mathbf{y}$) depend on β, we solve iteratively.

Special Cases

1. Canonical Link Function

If a canonical link is used, the estimating equations simplify to:

$$\mathbf{X}'\mathbf{y} = \mathbf{X}'$$

2. Identity Link Function

If the identity link is used, the estimating equation reduces to:

$$\mathbf{X}'\mathbf{V}^{-1}\mathbf{y} = \mathbf{X}'\mathbf{V}^{-1}\mathbf{X}$$

which leads to the Generalized Least Squares estimator:

$$\hat{\beta} = (\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{y}$$

Fisher-Scoring Algorithm in General Form

The iterative update formula for Fisher-scoring can be rewritten as:

$$\beta^{(m+1)} = \beta^{(m)} + (\hat{\mathbf{F}}'\hat{\mathbf{V}}^{-1}\hat{\mathbf{F}})^{-1}\hat{\mathbf{F}}'\hat{\mathbf{V}}^{-1}(\mathbf{y} - \hat{\ })$$

Since $\hat{F}, \hat{V}, \hat{\mu}$ depend on β, we evaluate them at $\beta^{(m)}$.

From an initial guess $\beta^{(0)}$, we iterate until convergence.

Notes:

- If $a(\phi)$ is a constant or takes the form $m_i\phi$ with known m_i, then ϕ cancels from the equations, simplifying the estimation.

7.8.7.2 Estimation of Dispersion Parameter (ϕ)

There are two common approaches to estimating ϕ:

1. Maximum Likelihood Estimation

The derivative of the log-likelihood with respect to ϕ is:

$$\frac{\partial l_i}{\partial \phi} = \frac{(\theta_i y_i - b(\theta_i)a'(\phi))}{a^2(\phi)} + \frac{\partial c(y_i, \phi)}{\partial \phi}$$

The MLE of ϕ satisfies the equation:

$$\frac{a^2(\phi)}{a'(\phi)} \sum_{i=1}^{n} \frac{\partial c(y_i, \phi)}{\partial \phi} = \sum_{i=1}^{n} (\theta_i y_i - b(\theta_i))$$

Challenges:

- For distributions other than the normal case, the expression for $\frac{\partial c(y,\phi)}{\partial \phi}$ is often complicated.
- Even with a **canonical link function** and constant $a(\phi)$, there is no simple general expression for the expected Fisher information:

$$-E\left(\frac{\partial^2 l}{\partial \phi^2}\right)$$

This means that the unification GLMs provide for estimating β does not extend as neatly to ϕ.

---

2. Moment Estimation (Bias-Corrected χ^2 Method)

- The MLE is **not** the conventional approach for estimating ϕ in Generalized Linear Models.
- For an exponential family distribution, the variance function is:

$$\text{Var}(Y) = V(\mu)a(\phi)$$

This implies the following moment-based estimator:

$$a(\phi) = \frac{\text{Var}(Y)}{V(\mu)} = \frac{E(Y-\mu)^2}{V(\mu)}$$
$$a(\hat{\phi}) = \frac{1}{n-p} \sum_{i=1}^{n} \frac{(y_i - \hat{\mu}_i)^2}{V(\hat{\mu}_i)}$$

where p is the number of parameters in β.

- For a **GLM with a canonical link function** $g(.) = (b'(.))^{-1}$:

$$g(\mu) = \theta = \eta = \mathbf{x}'$$
$$\mu = g^{-1}(\eta) = b'(\eta)$$

Using this, the moment estimator for $a(\phi) = \phi$ becomes:

$$\hat{\phi} = \frac{1}{n-p} \sum_{i=1}^{n} \frac{(y_i - g^{-1}(\hat{\eta}_i))^2}{V(g^{-1}(\hat{\eta}_i))}$$

Approach	Description	Pros	Cons
MLE	Estimates ϕ by maximizing the likelihood function	Theoretically optimal	Computationally complex, lacks a general closed-form solution
Moment Estimation	Uses a bias-corrected χ^2 method based on residual variance	Simpler, widely used in GLMs	Not as efficient as MLE

7.8.8 Inference

The estimated variance of $\hat{\beta}$ is given by:

$$\hat{\text{var}}(\beta) = a(\phi)(\hat{\mathbf{F}}'\hat{\mathbf{V}}^{-1}\hat{\mathbf{F}})^{-1}$$

where:

- $\mathbf{V}$ is an $n \times n$ diagonal matrix with diagonal elements given by $V(\mu_i)$.
- $\mathbf{F}$ is an $n \times p$ matrix given by:

$$\mathbf{F} = \frac{\partial \mu}{\partial \beta}$$

- Since $\mathbf{V}$ and $\mathbf{F}$ depend on the mean μ (and thus on β), their estimates $\hat{\mathbf{V}}$ and $\hat{\mathbf{F}}$ depend on $\hat{\beta}$.

To test a general hypothesis:

$$H_0 : \mathbf{L} \ = \mathbf{d}$$

where $\mathbf{L}$ is a $q \times p$ matrix, we use the **Wald test**:

$$W = (\mathbf{L}\hat{} \ - \mathbf{d})'(\mathbf{a}(\ )\mathbf{L}(\hat{\mathbf{F}}'\hat{\mathbf{V}}^{-1}\hat{\mathbf{F}})\mathbf{L}')^{-1}(\mathbf{L}\hat{} \ - \mathbf{d})$$

Under H_0, the Wald statistic follows a **chi-square** distribution:

$$W \sim \chi^2_q$$

where q is the rank of $\mathbf{L}$.

Special Case: Testing a Single Coefficient

For a hypothesis of the form:

$$H_0 : \beta_j = 0$$

the Wald test simplifies to:

$$W = \frac{\hat{\beta}_j^2}{\hat{\text{var}}(\hat{\beta}_j)} \sim \chi_1^2$$

asymptotically.

Another common test is the likelihood ratio test, which compares the likelihoods of a **full model** and a **reduced model**:

$$\Lambda = 2(l(\hat{\beta}_f) - l(\hat{\beta}_r)) \sim \chi_q^2$$

where:

- $l(\hat{\beta}_f)$ is the log-likelihood of the **full** model.
- $l(\hat{\beta}_r)$ is the log-likelihood of the **reduced** model.
- q is the number of constraints used in fitting the reduced model.

Test	Pros	Cons
Wald Test	Easy to compute, does not require fitting two models	May perform poorly in small samples
Likelihood Ratio Test	More accurate, especially for small samples	Requires fitting both full and reduced models

While the Wald test is more convenient, the likelihood ratio test is often preferred when sample sizes are small, as it tends to have better statistical properties.

7.8.9 Deviance

Deviance plays a crucial role in:

- **Goodness-of-fit assessment**: Checking how well a model explains the observed data.
- **Statistical inference**: Used in hypothesis testing, particularly likelihood ratio tests.

- **Estimating dispersion parameters**: Helps in refining variance estimates.
- **Model comparison**: Facilitates selection between competing models.

Assuming the dispersion parameter ϕ is known, let:

- $\tilde{\theta}$ be the **maximum likelihood estimate (MLE)** under the **full model**.
- $\hat{\theta}$ be the **MLE** under the **reduced model**.

The **likelihood ratio statistic** (twice the difference in log-likelihoods) is:

$$2\sum_{i=1}^{n} \frac{y_i(\tilde{\theta}_i - \hat{\theta}_i) - b(\tilde{\theta}_i) + b(\hat{\theta}_i)}{a_i(\phi)}$$

For **exponential family** distributions, the mean parameter is:

$$\mu = E(y) = b'(\theta)$$

Thus, the **natural parameter** is a function of μ:

$$\theta = \theta(\mu) = b'^{-1}(\mu)$$

Rewriting the likelihood ratio statistic in terms of μ:

$$2\sum_{i=1}^{n} \frac{y_i\{\theta(\tilde{\mu}_i) - \theta(\hat{\mu}_i)\} - b(\theta(\tilde{\mu}_i)) + b(\theta(\hat{\mu}_i))}{a_i(\phi)}$$

A **saturated model** provides the fullest possible fit, where each observation is perfectly predicted:

$$\tilde{\mu}_i = y_i, \quad i = 1, \dots, n$$

Setting $\tilde{\theta}_i^* = \theta(y_i)$ and $\hat{\theta}_i^* = \theta(\hat{\mu}_i)$, the likelihood ratio simplifies to:

$$2\sum_{i=1}^{n} \frac{y_i(\tilde{\theta}_i^* - \hat{\theta}_i^*) - b(\tilde{\theta}_i^*) + b(\hat{\theta}_i^*)}{a_i(\phi)}$$

Following McCullagh (2019), for $a_i(\phi) = \phi$, we define the **scaled deviance** as:

$$D^*(\mathbf{y}, \hat{\ }) = \frac{2}{\phi}\sum_{i=1}^{n}\{y_i(\tilde{\theta}_i^* - \hat{\theta}_i^*) - b(\tilde{\theta}_i^*) + b(\hat{\theta}_i^*)\}$$

and the **deviance** as:

$$D(\mathbf{y}, \hat{\ }) = \phi D^*(\mathbf{y}, \hat{\ })$$

where:

- $D^*(\mathbf{y}, \hat{\ })$ is **scaled deviance**.
- $D(\mathbf{y}, \hat{\ })$ is **deviance**.

In some models, $a_i(\phi) = \phi m_i$, where m_i is a known scalar that varies across observations. Then, the **scaled deviance** is:

$$D^*(\mathbf{y}, \hat{\ }) = \sum_{i=1}^{n} \frac{2\{y_i(\tilde{\theta}_i^* - \hat{\theta}_i^*) - b(\tilde{\theta}_i^*) + b(\hat{\theta}_i^*)\}}{\phi m_i}$$

The deviance is often decomposed into **deviance contributions**:

$$D^*(\mathbf{y}, \hat{\ }) = \sum_{i=1}^{n} d_i$$

where d_i is the **deviance contribution** from the i-th observation.

Uses of Deviance:

- D is used for model selection.
- D^* is used for goodness-of-fit tests, as it is a likelihood ratio statistic:

$$D^*(\mathbf{y}, \hat{\ }) = 2\{l(\mathbf{y}, \tilde{\ }) - l(\mathbf{y}, \hat{\ })\}$$

- The individual deviance contributions d_i are used to form deviance residuals.

Deviance for Normal Distribution

For the normal distribution:

$$\begin{aligned} \theta &= \mu, \\ \phi &= \sigma^2, \\ b(\theta) &= \frac{1}{2}\theta^2, \\ a(\phi) &= \phi. \end{aligned}$$

The MLEs are:

$$\begin{aligned}\tilde{\theta}_i &= y_i,\\ \hat{\theta}_i &= \hat{\mu}_i = g^{-1}(\hat{\eta}_i).\end{aligned}$$

The **deviance** simplifies to:

$$\begin{aligned}D &= 2\sum_{i=1}^{n}\left(y_i^2 - y_i\hat{\mu}_i - \frac{1}{2}y_i^2 + \frac{1}{2}\hat{\mu}_i^2\right)\\ &= \sum_{i=1}^{n}(y_i^2 - 2y_i\hat{\mu}_i + \hat{\mu}_i^2)\\ &= \sum_{i=1}^{n}(y_i - \hat{\mu}_i)^2.\end{aligned}$$

Thus, for the normal model, the deviance corresponds to the residual sum of squares.

Deviance for Poisson Distribution

For the Poisson distribution:

$$\begin{aligned}f(y) &= \exp\{y\log(\mu) - \mu - \log(y!)\},\\ \theta &= \log(\mu),\\ b(\theta) &= \exp(\theta),\\ a(\phi) &= 1.\end{aligned}$$

MLEs:

$$\begin{aligned}\tilde{\theta}_i &= \log(y_i),\\ \hat{\theta}_i &= \log(\hat{\mu}_i),\\ \hat{\mu}_i &= g^{-1}(\hat{\eta}_i).\end{aligned}$$

The **deviance** simplifies to:

$$\begin{aligned}D &= 2\sum_{i=1}^{n}(y_i\log(y_i) - y_i\log(\hat{\mu}_i) - y_i + \hat{\mu}_i)\\ &= 2\sum_{i=1}^{n}\left[y_i\log\left(\frac{y_i}{\hat{\mu}_i}\right) - (y_i - \hat{\mu}_i)\right].\end{aligned}$$

The **deviance contribution** for each observation:

$$d_i = 2\left[y_i \log\left(\frac{y_i}{\hat{\mu}_i}\right) - (y_i - \hat{\mu}_i)\right].$$

7.8.9.1 Analysis of Deviance

The **Analysis of Deviance** is a likelihood-based approach for comparing nested models in GLM. It is analogous to the [Analysis of Variance (ANOVA)] in linear models.

When comparing a **reduced model** (denoted by $\hat{\mu}_r$) and a **full model** (denoted by $\hat{\mu}_f$), the difference in **scaled deviance** follows an asymptotic chi-square distribution:

$$D^*(\mathbf{y}, \hat{\ }_{\mathbf{r}}) - D^*(\mathbf{y}, \hat{\ }_{\mathbf{f}}) = 2\{l(\mathbf{y}, \hat{\ }_{\mathbf{f}}) - l(\mathbf{y}, \hat{\ }_{\mathbf{r}})\} \sim \chi^2_q$$

where q is the difference in the number of free parameters between the two models.

This test provides a means to assess whether the additional parameters in the full model significantly improve model fit.

The dispersion parameter ϕ is estimated as:

$$\hat{\phi} = \frac{D(\mathbf{y}, \hat{\ })}{n - p}$$

where:

- $D(\mathbf{y}, \hat{\ })$ is the deviance of the fitted model.
- n is the total number of observations.
- p is the number of estimated parameters.

Caution: The frequent use of χ^2 tests can be problematic due to their reliance on asymptotic approximations, especially for small samples or overdispersed data (McCullagh, 2019).

7.8.9.2 Deviance Residuals

Since deviance plays a role in model evaluation, we define **deviance residuals** to examine individual data points. Given that the total deviance is:

$$D = \sum_{i=1}^{n} d_i$$

the **deviance residual** for observation i is:

$$r_{D_i} = \text{sign}(y_i - \hat{\mu}_i)\sqrt{d_i}$$

where:

- d_i is the deviance contribution from observation i.
- The **sign function** preserves the direction of the residual.

To account for varying leverage, we define the **standardized deviance residual**:

$$r_{s,i} = \frac{y_i - \hat{\mu}_i}{\hat{\sigma}(1 - h_{ii})^{1/2}}$$

where:

- h_{ii} is the **leverage** of observation i.
- $\hat{\sigma}$ is an estimate of the standard deviation.

Alternatively, using the **GLM hat matrix**:

$$\mathbf{H}^{GLM} = \mathbf{W}^{1/2} X(X'WX)^{-1} X'\mathbf{W}^{-1/2}$$

where $\mathbf{W}$ is an $n \times n$ diagonal matrix with (i,i)-th element given by w_i (see Estimation of Systematic Component (β)), we express standardized deviance residuals as:

$$r_{s,D_i} = \frac{r_{D_i}}{\{\hat{\phi}(1 - h_{ii}^{glm})\}^{1/2}}$$

where h_{ii}^{glm} is the i-th diagonal element of $\mathbf{H}^{GLM}$.

7.8.9.3 Pearson Chi-Square Residuals

Another goodness-of-fit statistic is the **Pearson Chi-Square** statistic, defined as:

$$X^2 = \sum_{i=1}^{n} \frac{(y_i - \hat{\mu}_i)^2}{V(\hat{\mu}_i)}$$

where:

- $\hat{\mu}_i$ is the fitted mean response.
- $V(\hat{\mu}_i)$ is the variance function of the response.

The **Scaled Pearson** χ^2 statistic is:

$$\frac{X^2}{\phi} \sim \chi^2_{n-p}$$

where p is the number of estimated parameters.

The **Pearson residuals** are:

$$X_i^2 = \frac{(y_i - \hat{\mu}_i)^2}{V(\hat{\mu}_i)}$$

These residuals assess the difference between observed and fitted values, standardized by the variance.

If the assumptions hold:

- **Independent samples**
- **No overdispersion**: If $\phi = 1$, we expect:

$$\frac{D(\mathbf{y}; \hat{\ })}{n-p} \approx 1, \quad \frac{X^2}{n-p} \approx 1$$

  A value **substantially greater than 1** suggests **overdispersion** or **model misspecification**.

- **Multiple groups**: If the model includes categorical predictors, separate calculations may be needed for each group.

Under these assumptions, both:

$$\frac{X^2}{\phi} \quad \text{and} \quad D^*(\mathbf{y}; \hat{\ })$$

follow a χ^2_{n-p} distribution.

```r
set.seed(123)
n <- 100
x <- rnorm(n)
y <- rpois(n, lambda = exp(0.5 + 0.3 * x))

# Poisson response
# Fit Poisson GLM
```

```r
fit <- glm(y ~ x, family = poisson(link = "log"))

# Extract deviance and Pearson residuals
deviance_residuals <- residuals(fit, type = "deviance")
pearson_residuals  <- residuals(fit, type = "pearson")
# Display residual summaries
summary(deviance_residuals)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> -2.1286 -1.0189 -0.1441 -0.1806  0.6171  1.8942
summary(pearson_residuals)
#>      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
#> -1.505136 -0.837798 -0.140873  0.002003  0.658076  2.366618
```

- The residuals suggest a reasonably **well-fitted model** with a **slight underprediction** tendency.
- No extreme deviations or severe skewness.
- The largest residuals indicate **potential outliers**, but they are not extreme enough to immediately suggest model inadequacy.

7.8.10 Diagnostic Plots

Standardized residual plots help diagnose potential issues with model specification, such as an incorrect link function or variance structure. Common residual plots include:

- **Plot of Standardized Deviance Residuals vs. Fitted Values**

  $$\text{plot}(r_{s,D_i}, \hat{\mu}_i)$$

  or

  $$\text{plot}(r_{s,D_i}, T(\hat{\mu}_i))$$

  where $T(\hat{\mu}_i)$ is a **constant information scale transformation**, which adjusts $\hat{\mu}_i$ to maintain a stable variance across observations.

- **Plot of Standardized Deviance Residuals vs. Linear Predictor**

  $$\text{plot}(r_{s,D_i}, \hat{\eta}_i)$$

  where $\hat{\eta}_i$ represents the linear predictor before applying the link function.

The following table summarizes the commonly used transformations $T(\hat{\mu}_i)$ for different random components:

Random Component	$T(\hat{\mu}_i)$
Normal	$\hat{\mu}$
Poisson	$2\sqrt{\hat{\mu}}$
Binomial	$2\sin^{-1}(\sqrt{\hat{\mu}})$
Gamma	$2\log(\hat{\mu})$
Inverse Gaussian	$-2\hat{\mu}^{-1/2}$

Interpretation of Residual Plots

- Trend in residuals: Indicates an incorrect link function or an inappropriate scale transformation.
- Systematic change in residual range with $T(\hat{\mu})$: Suggests an incorrect choice of random component—i.e., the assumed variance function does not match the data.
- Plot of absolute residuals vs. fitted values:

  $$\text{plot}(|r_{D_i}|, \hat{\mu}_i)$$

  This checks whether the variance function is correctly specified.

7.8.11 Goodness of Fit

To assess how well the model fits the data, we use:

- Deviance
- Pearson Chi-square Residuals

For **nested models**, likelihood-based information criteria provide a comparison metric:

$$\begin{aligned} AIC &= -2l(\hat{\ }) + 2p \\ AICC &= -2l(\hat{\ }) + 2p\left(\frac{n}{n-p-1}\right) \\ BIC &= -2l(\hat{\ }) + p\log(n) \end{aligned}$$

where:

- $l(\hat{\mu})$ is the log-likelihood at the estimated parameters.
- p is the number of model parameters.
- n is the number of observations.

Important Considerations:

- The **same data and model structure** (i.e., same link function and assumed random component distribution) must be used for comparisons.

- Models can differ in **number of parameters** but must remain consistent in other respects.

While traditional R^2 is not directly applicable to GLMs, an analogous measure is:

$$R_p^2 = 1 - \frac{l(\hat{\mu})}{l(\hat{\mu}_0)}$$

where $l(\hat{\mu}_0)$ is the log-likelihood of a model with only an intercept.

For binary response models, a **rescaled generalized** R^2 is often used:

$$\bar{R}^2 = \frac{R_*^2}{\max(R_*^2)} = \frac{1 - \exp\left(-\frac{2}{n}(l(\hat{\mu}) - l(\hat{\mu}_0))\right)}{1 - \exp\left(\frac{2}{n} l(\hat{\mu}_0)\right)}$$

where the denominator ensures the maximum possible R^2 scaling.

7.8.12 Over-Dispersion

Over-dispersion occurs when the observed variance exceeds what the assumed model allows. This is common in Poisson and Binomial models.

Variance Assumptions for Common Random Components

Random Component	Standard Assumption (var(Y))	Alternative Model Allowing Over-Dispersion ($V(\mu)$)
Binomial	$n\mu(1-\mu)$	$\phi n\mu(1-\mu)$, where $m_i = n$
Poisson	μ	$\phi\mu$

- **By default,** $\phi = 1$, meaning the variance follows the assumed model.
- If $\phi \neq 1$, the variance differs from the expectation:
  - $\phi > 1$: **Over-dispersion** (greater variance than expected).
  - $\phi < 1$: **Under-dispersion** (less variance than expected).

This discrepancy suggests the assumed **random component distribution** may be inappropriate.

To account for dispersion issues, we can:

1. **Choose a more flexible random component distribution**
   - Negative Binomial for over-dispersed Poisson data.
   - Conway-Maxwell Poisson for both over- and under-dispersed count data.
2. **Use Mixed Models to account for random effects**

- Nonlinear and Generalized Linear Mixed Models introduce random effects, which can help capture additional variance.

```r
# Load necessary libraries
library(ggplot2)
library(MASS)   # For negative binomial models
library(MuMIn)  # For R^2 in GLMs
library(glmmTMB) # For handling overdispersion

# Generate Example Data
set.seed(123)
n <- 100
x <- rnorm(n)
mu <- exp(0.5 + 0.3 * x)  # True mean function
y <- rpois(n, lambda = mu)  # Poisson outcome

# Fit a Poisson GLM
model_pois <- glm(y ~ x, family = poisson(link = "log"))

# Compute residuals
resid_dev <- residuals(model_pois, type = "deviance")
fitted_vals <- fitted(model_pois)

# Standardized Residual Plot: Residuals vs Fitted Values
ggplot(data = data.frame(fitted_vals, resid_dev),
       aes(x = fitted_vals, y = resid_dev)) +
    geom_point(alpha = 0.6) +
    geom_smooth(method = "loess",
                color = "red",
                se = FALSE) +
    theme_minimal() +
    labs(title = "Standardized Deviance Residuals vs Fitted Values",
         x = "Fitted Values", y = "Standardized Deviance Residuals")
```

```r
# Absolute Residuals vs Fitted Values (Variance Function Check)
ggplot(data = data.frame(fitted_vals, abs_resid = abs(resid_dev)),
       aes(x = fitted_vals, y = abs_resid)) +
    geom_point(alpha = 0.6) +
    geom_smooth(method = "loess",
                color = "blue",
                se = FALSE) +
    theme_minimal() +
    labs(title = "Absolute Deviance Residuals vs Fitted Values",
         x = "Fitted Values", y = "|Residuals|")
```

```
# Goodness-of-Fit Metrics
AIC(model_pois)     # Akaike Information Criterion
#> [1] 322.9552
BIC(model_pois)     # Bayesian Information Criterion
#> [1] 328.1655
logLik(model_pois) # Log-likelihood
#> 'log Lik.' -159.4776 (df=2)

# R-squared for GLM
r_squared <-
    1 - (logLik(model_pois) / logLik(glm(y ~ 1, family = poisson)))
r_squared
#> 'log Lik.' 0.05192856 (df=2)

# Overdispersion Check
resid_pearson <- residuals(model_pois, type = "pearson")
# Estimated dispersion parameter
phi_hat <-
    sum(resid_pearson ^ 2) / (n - length(coef(model_pois)))
phi_hat
#> [1] 1.055881

# If phi > 1,
# fit a Negative Binomial Model to correct for overdispersion
if (phi_hat > 1) {
    model_nb <- glm.nb(y ~ x)
```

```
    summary(model_nb)
}
#>
#> Call:
#> glm.nb(formula = y ~ x, init.theta = 50.70707605, link = log)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -2.1034  -0.9978  -0.1393   0.6047   1.8553
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  0.42913    0.08514   5.040 4.65e-07 ***
#> x            0.35262    0.08654   4.075 4.61e-05 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for Negative Binomial(50.7071) family taken to be 1)
#>
#>     Null deviance: 135.71  on 99  degrees of freedom
#> Residual deviance: 118.92  on 98  degrees of freedom
#> AIC: 324.91
#>
#> Number of Fisher Scoring iterations: 1
#>
#>
#>               Theta:  51
#>           Std. Err.:  233
#>
#>  2 x log-likelihood:  -318.906
```

1. Standardized Residuals vs. Fitted Values

- If there is a clear trend in the residual plot (e.g., a curve), this suggests an incorrect link function.
- If the residual spread increases with fitted values, the variance function may be misspecified.

2. Absolute Residuals vs. Fitted Values

- A systematic pattern (e.g., funnel shape) suggests heteroscedasticity (i.e., changing variance).
- If the residuals increase with fitted values, a different variance function (e.g., Negative Binomial instead of Poisson) may be needed.

3. Goodness-of-Fit Metrics

- AIC/BIC: Lower values indicate a better model, but must be compared across nested models.
- Log-likelihood: Higher values suggest a better-fitting model.
- R^2 for GLMs: Since traditional R^2 is not available, the likelihood-based R^2 is used to measure improvement over a null model.

4. Overdispersion Check (ϕ)

- If $\phi \approx 1$, the Poisson assumption is valid.
- If $\phi > 1$, there is overdispersion, meaning a Negative Binomial model may be more appropriate.
- If $\phi < 1$, underdispersion is present, requiring alternative distributions like the Conway-Maxwell Poisson.

Chapter 8

Linear Mixed Models

8.1 Dependent Data

In many real-world applications, observations are not independent but exhibit **correlations** due to shared characteristics. Below are common forms of dependent data:

- **Multivariate measurements on individuals**: Multiple attributes measured on the same person may be correlated (e.g., blood pressure, cholesterol, and BMI).
- **Clustered measurements**: Individuals within a shared environment (e.g., families, schools, hospitals) often exhibit correlated responses.
- **Repeated measurements**: When the same individual is measured multiple times, correlations arise naturally.
  - *Example*: Tracking cholesterol levels of a person over time.
  - If these repeated measurements follow an experimental design where treatments were applied initially, they are referred to as repeated measures (Schabenberger and Pierce, 2001).
- **Longitudinal data**: Repeated measurements taken over time in an observational study are known as longitudinal data (Schabenberger and Pierce, 2001).
- **Spatial data**: Measurements taken from individuals in nearby locations (e.g., residents of the same neighborhood) often exhibit spatial correlation.

Since standard linear regression assumes independent observations, these correlations violate its assumptions. Thus, Linear Mixed Models (LMMs) provide a framework to account for these dependencies.

A Linear Mixed Model (also called a **Mixed Linear Model**) consists of two components:

- **Fixed effects**: Parameters associated with variables that have the same

effect across all observations (e.g., gender, age, diet, time).
- **Random effects**: Individual-specific variations or correlation structures (e.g., subject-specific effects, spatial correlations), leading to dependent (correlated) errors.

A key advantage of LMMs is that they model random subject-specific effects, rather than including individual dummy variables. This provides:

- A reduction in the number of parameters to estimate, avoiding overfitting.
- A framework for inference on a population level, rather than restricting conclusions to observed individuals.

8.1.1 Motivation: A Repeated Measurements Example

Consider a scenario where we analyze **repeated measurements** of a response variable Y_{ij} for the i-th subject at time j:

- $i = 1, \dots, N$ (subjects)
- $j = 1, \dots, n_i$ (measurements per subject)

We define the response vector for subject i as:

$$\mathbf{Y}_i = \begin{bmatrix} Y_{i1} \\ Y_{i2} \\ \vdots \\ Y_{in_i} \end{bmatrix}$$

To model this data, we use a two-stage hierarchical approach:

8.1.1.1 Stage 1: Regression Model (Within-Subject Variation) {#sec-stage-1-regression-model-(within-subject-variation}

We first model how the response changes over time for each subject:

$$\mathbf{Y}_i = Z_i \beta_i + \epsilon_i$$

where:

- Z_i is an $n_i \times q$ matrix of known covariates (e.g., time, treatment).
- β_i is a $q \times 1$ vector of subject-specific regression coefficients.
- ϵ_i represents random errors, often assumed to follow $\epsilon_i \sim N(0, \sigma^2 I)$.

At this stage, each individual has their own unique regression coefficients β_i. However, estimating a separate β_i for each subject is impractical when N is large. Thus, we introduce a [second stage](#sec-stage-2-parameter-model-(between-subject-variation) to impose structure on β_i.

8.1.1.2 Stage 2: Parameter Model (Between-Subject Variation) {#sec-stage-2-parameter-model-(between-subject-variation}

We assume that the subject-specific coefficients β_i arise from a common population distribution:

$$\beta_i = K_i\beta + b_i$$

where:

- K_i is a $q \times p$ matrix of known covariates.
- β is a $p \times 1$ vector of global parameters (fixed effects).
- $\mathbf{b}_i$ are random effects, assumed to follow $\mathbf{b}_i \sim N(0, D)$.

Thus, each individual's regression coefficients β_i are modeled as deviations from a population-level mean β, with subject-specific variations b_i. This hierarchical structure enables:

- Borrowing of strength: Individual estimates β_i are informed by the overall population distribution.
- Improved generalization: The model captures both fixed and random variability efficiently.

The full Linear Mixed Model can then be written as:

$$\mathbf{Y}_i = Z_iK_i\beta + Z_ib_i + \epsilon_i$$

where:

- $Z_iK_i\beta$ represents the **fixed effects** component.
- Z_ib_i represents the **random effects** component.
- ϵ_i represents the residual errors.

This formulation accounts for both **within-subject correlations** (via random effects) and **between-subject variability** (via fixed effects), making it a powerful tool for analyzing dependent data.

8.1.2 Example: Linear Mixed Model for Repeated Measurements

8.1.2.1 Stage 1: Subject-Specific Model

The first stage models the response variable Y_{ij} for subject i at time t_{ij}:

$$Y_{ij} = \beta_{1i} + \beta_{2i}t_{ij} + \epsilon_{ij}$$

where:

- $j = 1, \dots, n_i$ represents different time points for subject i.
- β_{1i} is the subject-specific intercept (baseline response for subject i).
- β_{2i} is the subject-specific slope (rate of change over time).
- $\epsilon_{ij} \sim N(0, \sigma^2)$ are independent errors.

In matrix notation, the model is written as:

$$\mathbf{Y}_i = \begin{bmatrix} Y_{i1} \\ Y_{i2} \\ \vdots \\ Y_{in_i} \end{bmatrix}, \quad \mathbf{Z}_i = \begin{bmatrix} 1 & t_{i1} \\ 1 & t_{i2} \\ \vdots & \vdots \\ 1 & t_{in_i} \end{bmatrix},$$

$$\beta_i = \begin{bmatrix} \beta_{1i} \\ \beta_{2i} \end{bmatrix}, \quad \epsilon_i = \begin{bmatrix} \epsilon_{i1} \\ \epsilon_{i2} \\ \vdots \\ \epsilon_{in_i} \end{bmatrix}.$$

Thus, the model can be rewritten as:

$$\mathbf{Y_i} = \mathbf{Z_i}\ _{\mathbf{i}} + \ _{\mathbf{i}}.$$

8.1.2.2 Stage 2: Population-Level Model

Since estimating a separate β_{1i} and β_{2i} for each subject is impractical, we assume they follow a population distribution:

$$\begin{aligned} \beta_{1i} &= \beta_0 + b_{1i}, \\ \beta_{2i} &= \beta_1 L_i + \beta_2 H_i + \beta_3 C_i + b_{2i}. \end{aligned}$$

where:

- L_i, H_i, C_i are indicator variables for treatment groups:
  - $L_i = 1$ if the subject belongs to treatment group 1, else 0.
  - $H_i = 1$ if the subject belongs to treatment group 2, else 0.
  - $C_i = 1$ if the subject belongs to treatment group 3, else 0.
- β_0 represents the average baseline response across all subjects.
- $\beta_1, \beta_2, \beta_3$ are the average time effects for the respective treatment groups.
- b_{1i}, b_{2i} are random effects representing subject-specific deviations.

This structure implies that while the intercept β_{1i} varies randomly across subjects, the slope β_{2i} depends both on treatment and random subject-specific deviations.

In matrix form, this is:

$$\mathbf{K}_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & L_i & H_i & C_i \end{bmatrix},$$
$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix},$$
$$\mathbf{b}_i = \begin{bmatrix} b_{1i} \\ b_{2i} \end{bmatrix},$$
$$\beta_i = \mathbf{K_i} + \mathbf{b_i}.$$

8.1.2.3 Final Mixed Model Formulation

Substituting Stage 2 into Stage 1, we obtain the full mixed model:

$$\mathbf{Y}_i = \mathbf{Z}_i(\mathbf{K}_i\beta + \mathbf{b}_i) + \epsilon_i.$$

Expanding:

$$\mathbf{Y}_i = \mathbf{Z}_i\mathbf{K}_i\beta + \mathbf{Z}_i\mathbf{b}_i + \epsilon_i.$$

Interpretation:

- $\mathbf{Z}_i\mathbf{K}_i\beta$ represents the **fixed effects** (population-level trends).
- $\mathbf{Z}_i\mathbf{b}_i$ represents the **random effects** (subject-specific variations).
- ϵ_i represents the **residual errors**.

Assumptions:

- Random effects: $\mathbf{b}_i \sim N(0, D)$, where D is the variance-covariance matrix.
- Residual errors: $\epsilon_i \sim N(0, \sigma^2 I)$.
- Independence: $\mathbf{b}_i$ and ϵ_i are independent.

To estimate $\hat{\beta}$, one might consider a sequential approach:

1. Estimate $\hat{\beta}_i$ for each subject in the first stage.
2. Estimate $\hat{\beta}$ in the second stage by replacing β_i with $\hat{\beta}_i$.

However, this method introduces several problems:

- Loss of information: Summarizing $\mathbf{Y}_i$ solely by $\hat{\beta}_i$ discards valuable within-subject variability.
- Ignoring uncertainty: Treating estimated values $\hat{\beta}_i$ as known values leads to underestimated variability.

- Unequal sample sizes: Subjects may have different numbers of observations (n_i), which affects variance estimation.

To address these issues, we adopt the Linear Mixed Model (LMM) framework (Laird and Ware, 1982).

8.1.2.4 Reformulating the Linear Mixed Model

Substituting [Stage 2](#sec-stage-2-parameter-model-(between-subject-variation) into [Stage 1](#sec-stage-2-parameter-model-(between-subject-variation), we obtain:

$$\mathbf{Y}_i = \mathbf{Z}_i\mathbf{K}_i\beta + \mathbf{Z}_i\mathbf{b}_i + \ _i.$$

Defining $\mathbf{X}_i = \mathbf{Z}_i\mathbf{K}_i$ as an $n_i \times p$ matrix of covariates, we rewrite the model as:

$$\mathbf{Y}_i = \mathbf{X}_i\beta + \mathbf{Z}_i\mathbf{b}_i + \ _i.$$

where:

- $i = 1, \dots, N$ (subjects).
- β are **fixed effects**, common to all subjects.
- $\mathbf{b}_i$ are **subject-specific random effects**, assumed to follow:

$$\mathbf{b}_i \sim N_q(\mathbf{0}, \mathbf{D}).$$

- $_i$ represents residual errors:

$$_i \sim N_{n_i}(\mathbf{0}, \ _{\mathbf{i}}).$$

- Independence assumption: $\mathbf{b}_i$ and $_i$ are independent.
- Dimension notation:
  - $\mathbf{Z}_i$ is an $n_i \times q$ matrix of known covariates for random effects.
  - $\mathbf{X}_i$ is an $n_i \times p$ matrix of known covariates for fixed effects.

8.1.2.5 Hierarchical (Conditional) Formulation

Rewriting the LMM in **hierarchical form**:

$$\begin{aligned}\mathbf{Y}_i|\mathbf{b}_i &\sim N(\mathbf{X}_i\beta + \mathbf{Z}_i\mathbf{b}_i,\ _i),\\ \mathbf{b}_i &\sim N(\mathbf{0}, \mathbf{D}).\end{aligned}$$

where:

- The first equation states that, given the subject-specific random effects $\mathbf{b}_i$, the response follows a normal distribution with mean $\mathbf{X}_i\beta + \mathbf{Z}_i\mathbf{b}_i$ and covariance $_i$.
- The second equation states that the random effects $\mathbf{b}_i$ follow a **multivariate normal** distribution with mean zero and covariance D.

We denote the respective probability density functions as:

$$f(\mathbf{Y}_i|\mathbf{b}_i) \quad \text{and} \quad f(\mathbf{b}_i).$$

Using the general **marginalization formula**:

$$\begin{aligned}f(A, B) &= f(A|B)f(B)\\ f(A) &= \int f(A, B)dB = \int f(A|B)f(B)dB\end{aligned}$$

we obtain the **marginal density** of $\mathbf{Y}_i$:

$$f(\mathbf{Y}_i) = \int f(\mathbf{Y}_i|\mathbf{b}_i)f(\mathbf{b}_i)d\mathbf{b}_i.$$

Solving this integral, we obtain:

$$\mathbf{Y}_i \sim N(\mathbf{X_i}\ , \mathbf{Z_i}\mathbf{D}\mathbf{Z_i'} +\ _\mathbf{i}).$$

Interpretation:

- **Mean structure:** The expectation remains $\mathbf{X}_i\beta$, the fixed effects.
- **Variance structure:** The covariance now incorporates random effect variability:

  $$\mathbf{Z_i}\mathbf{D}\mathbf{Z_i'} +\ _\mathbf{i}.$$

  This shows that the random effects contribute additional correlation between observations.

Key Takeaway:
The marginal formulation of LMM no longer includes $Z_i b_i$ in the mean, but instead incorporates it into the covariance structure, introducing marginal dependence in $\mathbf{Y}_i$.

Technically, rather than "averaging out" the random effect b_i, we add its contribution to the variance-covariance matrix.

Continue with the example:

$$Y_{ij} = (\beta_0 + b_{1i}) + (\beta_1 L_i + \beta_2 H_i + \beta_3 C_i + b_{2i}) t_{ij} + \epsilon_{ij}.$$

For each treatment group:

$$Y_{ij} = \begin{cases} \beta_0 + b_{1i} + (\beta_1 + b_{2i}) t_{ij} + \epsilon_{ij}, & L_i = 1 \\ \beta_0 + b_{1i} + (\beta_2 + b_{2i}) t_{ij} + \epsilon_{ij}, & H_i = 1 \\ \beta_0 + b_{1i} + (\beta_3 + b_{2i}) t_{ij} + \epsilon_{ij}, & C_i = 1. \end{cases}$$

Interpretation:

- **Intercepts and slopes are subject-specific**: Each subject has their own baseline response and rate of change.
- **Treatment groups affect slopes, but not intercepts**:
  - β_0 is the common intercept across groups.
  - Slopes differ by treatment: β_1 for L, β_2 for H, β_3 for C.
- **Random effects** introduce within-subject correlation:
  - b_{1i} allows individual variation in baseline response.
  - b_{2i} allows subject-specific deviations in slopes.

In the **hierarchical model form**, we again express the LMM as:

$$\begin{aligned} \mathbf{Y}_i | \mathbf{b}_i &\sim N(\mathbf{X}_i \beta + \mathbf{Z}_i \mathbf{b}_i, \ _i) \\ \mathbf{b}_i &\sim N(\mathbf{0}, \mathbf{D}) \end{aligned}$$

where:

- $\mathbf{X}_i$ and $\mathbf{Z}_i$ are design matrices for **fixed** and **random** effects, respectively.
- $\mathbf{b}_i$ represents **subject-specific random effects**.
- $_i$ is the **residual error covariance matrix**.
- $\mathbf{D}$ is the **random effects covariance matrix**.

The fixed-effects parameter vector is:

$$\beta = (\beta_0, \beta_1, \beta_2, \beta_3)'.$$

From the model structure, we define:

$$\mathbf{X}_i = \mathbf{Z}_i \mathbf{K}_i.$$

Expanding,

$$\mathbf{Z}_i = \begin{bmatrix} 1 & t_{i1} \\ 1 & t_{i2} \\ \vdots & \vdots \\ 1 & t_{in_i} \end{bmatrix}, \quad \mathbf{K}_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & L_i & H_i & C_i \end{bmatrix}.$$

Multiplying:

$$\mathbf{X}_i = \begin{bmatrix} 1 & t_{i1}L_i & t_{i1}H_i & t_{i1}C_i \\ 1 & t_{i2}L_i & t_{i2}H_i & t_{i2}C_i \\ \vdots & \vdots & \vdots & \vdots \\ 1 & t_{in_i}L_i & t_{in_i}H_i & t_{in_i}C_i \end{bmatrix}.$$

The random effects vector is:

$$\mathbf{b}_i = \begin{bmatrix} b_{1i} \\ b_{2i} \end{bmatrix}.$$

The covariance matrix $\mathbf{D}$ for random effects is:

$$\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} \\ d_{12} & d_{22} \end{bmatrix}.$$

We assume **conditional independence**:

$$_i = \sigma^2 \mathbf{I}_{n_i}.$$

This means that, **given the random effects $\mathbf{b}_i$** and β, the responses for subject i are independent.

To derive the **marginal model**, we integrate out the random effects $\mathbf{b}_i$. This leads to:

$$Y_{ij} = \beta_0 + \beta_1 L_i t_{ij} + \beta_2 H_i t_{ij} + \beta_3 C_i t_{ij} + \eta_{ij},$$

where:

$$\eta_i \sim N(\mathbf{0}, \mathbf{Z}_i \mathbf{D} \mathbf{Z}_i' + \ _i).$$

Thus, the full marginal model is:

$$\mathbf{Y_i} \sim N(\mathbf{X}_i \beta, \mathbf{Z}_i \mathbf{D} \mathbf{Z}_i' + \ _i).$$

Example: Case Where $n_i = 2$

For a subject with **two observations**, we compute:

$$\mathbf{Z}_i = \begin{bmatrix} 1 & t_{i1} \\ 1 & t_{i2} \end{bmatrix}.$$

Then:

$$\mathbf{Z}_i \mathbf{D} \mathbf{Z}_i' = \begin{bmatrix} 1 & t_{i1} \\ 1 & t_{i2} \end{bmatrix} \begin{bmatrix} d_{11} & d_{12} \\ d_{12} & d_{22} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ t_{i1} & t_{i2} \end{bmatrix}.$$

Expanding the multiplication:

$$\mathbf{Z}_i \mathbf{D} \mathbf{Z}_i' = \begin{bmatrix} d_{11} + 2d_{12}t_{i1} + d_{22}t_{i1}^2 & d_{11} + d_{12}(t_{i1} + t_{i2}) + d_{22}t_{i1}t_{i2} \\ d_{11} + d_{12}(t_{i1} + t_{i2}) + d_{22}t_{i1}t_{i2} & d_{11} + 2d_{12}t_{i2} + d_{22}t_{i2}^2 \end{bmatrix}.$$

Finally, incorporating the residual variance:

$$\mathrm{Var}(Y_{i1}) = d_{11} + 2d_{12}t_{i1} + d_{22}t_{i1}^2 + \sigma^2.$$

Interpretation of the Marginal Model

1. **Correlation in the Errors**:
   - The off-diagonal elements of $\mathbf{Z}_i \mathbf{D} \mathbf{Z}_i'$ introduce **correlation** between observations within the same subject.
   - This accounts for the fact that repeated measurements on the same individual are not independent.
2. **Variance Structure**:
   - The variance function of Y_{ij} is quadratic in time:

     $$\mathrm{Var}(Y_{ij}) = d_{11} + 2d_{12}t_{ij} + d_{22}t_{ij}^2 + \sigma^2.$$

   - The curvature of this variance function is determined by d_{22}.
   - If $d_{22} > 0$, variance increases over time.

Key Takeaway:
In the hierarchical model, random effects contribute to the mean structure.
In the marginal model, they affect the covariance structure, leading to heteroskedasticity (changing variance over time) and correlation between repeated measures.

8.1.3 Random-Intercepts Model

The Random-Intercepts Model is obtained by **removing random slopes**, meaning:

- All subject-specific variability in slopes is attributed only to treatment differences.
- The model allows each subject to have their own intercept, but within each treatment group, all subjects share the same slope.

The hierarchical (conditional) model is:

$$\begin{aligned} \mathbf{Y}_i | b_i &\sim N(\mathbf{X}_i\beta + 1b_i,\ _i), \\ b_i &\sim N(0, d_{11}). \end{aligned}$$

where:

- b_i is the **random intercept** for subject i, assumed to follow $N(0, d_{11})$.
- $\mathbf{X}_i$ contains the **fixed effects**, which include treatment and time.
- $_i$ represents **residual variance**, typically assumed to be $\sigma^2\mathbf{I}$ (independent errors).

Since there are no **random slopes**, the only source of **subject-specific variability** is the intercept.

Integrating out b_i and assuming conditional independence $_i = \sigma^2\mathbf{I}_{n_i}$, the **marginal distribution** of $\mathbf{Y}_i$ is:

$$\mathbf{Y}_i \sim N(\mathbf{X}_i\beta, 11'd_{11} + \sigma^2\mathbf{I}).$$

The **marginal covariance matrix** is:

$$\begin{aligned} \mathrm{Cov}(\mathbf{Y}_i) &= 11'd_{11} + \sigma^2 I \\ &= \begin{bmatrix} d_{11}+\sigma^2 & d_{11} & d_{11} & \dots & d_{11} \\ d_{11} & d_{11}+\sigma^2 & d_{11} & \dots & d_{11} \\ d_{11} & d_{11} & d_{11}+\sigma^2 & \dots & d_{11} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{11} & d_{11} & d_{11} & \dots & d_{11}+\sigma^2 \end{bmatrix}. \end{aligned}$$

The **correlation matrix** is obtained by standardizing the covariance matrix:

$$\text{Corr}(\mathbf{Y}_i) = \begin{bmatrix} 1 & \rho & \rho & \dots & \rho \\ \rho & 1 & \rho & \dots & \rho \\ \rho & \rho & 1 & \dots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \dots & 1 \end{bmatrix},$$

where:

$$\rho = \frac{d_{11}}{d_{11} + \sigma^2}.$$

This **correlation structure** is known as **compound symmetry**, meaning:

- **Constant variance**: $\text{Var}(Y_{ij}) = d_{11} + \sigma^2$ for all j.
- **Equal correlation**: $\text{Corr}(Y_{ij}, Y_{ik}) = \rho$ for all $j \neq k$.
- **Positive correlation**: Any two observations from the same subject are equally correlated.

Interpretation of ρ (Intra-Class Correlation)

- ρ is called the intra-class correlation coefficient (ICC).
- It measures the proportion of total variability that is due to between-subject variability:

$$\rho = \frac{\text{Between-Subject Variance}}{\text{Total Variance}} = \frac{d_{11}}{d_{11} + \sigma^2}.$$

- If ρ is large:
  - The inter-subject variability (d_{11}) is large relative to the intra-subject variability (σ^2).
  - This means subjects differ substantially in their intercepts.
  - Responses from the same subject are highly correlated.
- If ρ is small:
  - The residual error variance dominates, meaning individual differences in intercepts are small.
  - Measurements from the same subject are weakly correlated.

Summary of the Random-Intercepts Model

Feature	Explanation
Intercepts	Random, subject-specific (b_i)

Feature	Explanation
Slopes	Fixed within each treatment group
Covariance Structure	**Compound symmetry** (constant variance, equal correlation)
Intra-Class Correlation (ρ)	Measures **between-subject variability** relative to total variability
Interpretation	Large $\rho \to$ Strong subject-level differences, Small $\rho \to$ Mostly residual noise

---

8.1.4 Covariance Models in Linear Mixed Models

Previously, we assumed that the **within-subject errors** are **conditionally independent**, meaning:

$$_i = \sigma^2 \mathbf{I}_{n_i}.$$

However, real-world data often exhibit **correlated errors**, particularly in **longitudinal studies** where observations over time are influenced by underlying stochastic processes.

To model this dependence, we decompose the error term into two components:

$$\epsilon_i = \epsilon_{(1)i} + \epsilon_{(2)i},$$

where:

- $\epsilon_{(1)i}$ (Serial Correlation Component):
  - Represents subject-specific response to time-varying stochastic processes.
  - Captures dependency across observations for the same subject.
- $\epsilon_{(2)i}$ (Measurement Error Component):
  - Represents pure measurement error, assumed independent of $\epsilon_{(1)i}$.

Thus, the full LMM formulation becomes:

$$\mathbf{Y_i} = \mathbf{X_i} \ + \mathbf{Z_i b_i} + \ _{\mathbf{(1)i}} + \ _{\mathbf{(2)i}}.$$

where:

- Random effects: $\mathbf{b_i} \sim N(\mathbf{0}, \mathbf{D})$.
- Measurement errors: $\epsilon_{(2)i} \sim N(\mathbf{0}, \ ^{\mathbf{2}}\mathbf{I}_{\mathbf{n_i}})$.
- Serial correlation errors: $\epsilon_{(1)i} \sim N(\mathbf{0}, \ ^{\mathbf{2}}\mathbf{H_i})$.
- Independence assumption: $\mathbf{b}_i$ and ϵ_i are mutually independent.

To model the correlation structure of the **serial correlation component** $\epsilon_{(1)i}$, we define the $n_i \times n_i$ correlation (or covariance) matrix $\mathbf{H}_i$.

The (j, k)th element of $\mathbf{H}_i$ is denoted as:

$$h_{ijk} = g(t_{ij}, t_{ik}),$$

where:

- h_{ijk} represents the covariance (or correlation) between time points t_{ij} and t_{ik}.
- $g(t_{ij}, t_{ik})$ is a decreasing function that defines the correlation between time points t_{ij} and t_{ik}.
- Typically, we assume that this function depends only on the time difference (stationarity assumption):

  $$h_{ijk} = g(|t_{ij} - t_{ik}|)$$

  meaning that the correlation only depends on the absolute time lag.
- To be a valid correlation matrix, we require:

  $$g(0) = 1.$$

  This ensures that each observation has a perfect correlation with itself.

Common Choices for $g(|t_{ij} - t_{ik}|)$

Several functions can be used to define the **decay in correlation** as time differences increase.

1. Exponential Correlation (Continuous-Time AR(1))

$$g(|t_{ij} - t_{ik}|) = \exp(-\phi|t_{ij} - t_{ik}|)$$

- **Decay rate:** Controlled by $\phi > 0$.
- **Interpretation:**
  - Observations closer in time are more correlated.
  - The correlation decreases exponentially as time separation increases.
- **Use case:** Often used in biological and economic time-series models.

---

2. Gaussian Correlation (Squared Exponential Kernel)

$$g(|t_{ij} - t_{ik}|) = \exp(-\phi(t_{ij} - t_{ik})^2)$$

- **Decay rate:** Faster than exponential.
- **Interpretation:**
  - If ϕ is large, correlation drops sharply as time separation increases.
  - Produces smooth correlation structures.
- **Use case:** Used in spatial statistics and machine learning (Gaussian processes).

---

3. Autoregressive (AR(1)) Correlation

A **First-Order Autoregressive Model (AR(1))** assumes that the value at time t depends on its previous value:

$$\alpha_t = \phi\alpha_{t-1} + \eta_t,$$

where

- $\eta_t \sim$ i.i.d. $N(0, \sigma_\eta^2)$ (white noise process).
- ϕ is the **autocorrelation coefficient**.

Then, the covariance between two observations at different times is:

$$\mathrm{Cov}(\alpha_t, \alpha_{t+h}) = \frac{\sigma_\eta^2 \phi^{|h|}}{1 - \phi^2}.$$

Thus, the **correlation** between time points t and $t + h$ is:

$$\mathrm{Corr}(\alpha_t, \alpha_{t+h}) = \phi^{|h|}.$$

For a sequence of T time points, the resulting **Toeplitz correlation matrix** is:

$$\mathrm{Corr}(\alpha_T) = \begin{bmatrix} 1 & \phi^1 & \phi^2 & \dots & \phi^{T-1} \\ \phi^1 & 1 & \phi^1 & \dots & \phi^{T-2} \\ \phi^2 & \phi^1 & 1 & \dots & \phi^{T-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi^{T-1} & \phi^{T-2} & \phi^{T-3} & \dots & 1 \end{bmatrix}.$$

- **Decay rate:** ϕ controls how fast correlation decreases.
- **Use case:**
  - Common in **time series models**.
  - Appropriate when **observations are equally spaced in time**.

Properties of the AR(1) Structure:

- Correlation decreases with time lag:
  - Observations closer in time are more correlated.
  - Decay rate is controlled by ϕ.
- Toeplitz structure:
  - The covariance matrix exhibits a banded diagonal pattern.
  - Useful in longitudinal and time-series data.
- Applicability:
  - Small ϕ (≈ 0) $\rightarrow$ Weak autocorrelation (errors are mostly independent).
  - Large ϕ (≈ 1) $\rightarrow$ Strong autocorrelation (highly dependent observations).

8.1.5 Covariance Structures in Mixed Models

Structure	Covariance Function $g(\lvert t_{ij} - t_{ik}\rvert)$	Decay Behavior	Correlation Matrix Pattern	Key Properties	Common Use Cases
Compound Symmetry	Constant: $g(\lvert t_{ij} - t_{ik}\rvert) = \rho$	No decay	Constant off-diagonal	Equal correlation across all time points (homogeneous)	Random-intercepts model, repeated measures
AR(1) (Autoregressive)	$\phi^{\lvert t_{ij} - t_{ik}\rvert}$	Exponential decay	Toeplitz (banded diagonal)	Strong correlation for nearby observations, weak for distant ones	Time series, longitudinal data
Exponential	$\exp(-\phi\lvert t_{ij} - t_{ik}\rvert)$	Smooth decay	Spatially continuous	Models continuous correlation decline over time/space	Growth curves, ecological models
Gaussian (Squared Exponential)	$\exp(-\phi(t_{ij} - t_{ik})^2)$	Rapid decay	Spatially continuous	Smooth decay but stronger locality effect	Spatial processes, geostatistics

Structure	Covariance Function $g(\|t_{ij} - t_{ik}\|)$	Decay Behavior	Correlation Matrix Pattern	Key Properties	Common Use Cases
Toeplitz	Varies by lag, $g(\|t_{ij} - t_{ik}\|) = c_h$	General pattern	General symmetric matrix	Arbitrary structure, allows irregular spacing	Irregular time points, flexible spatial models
Unstructured	Fully parameterized, no constraints	No fixed pattern	General symmetric matrix	Allows any correlation pattern, but many parameters	Small datasets with unknown correlation
Banded	Zero correlation beyond a certain lag	Abrupt cut-off	Banded diagonal	Assumes only close observations are correlated	Large datasets, high-dimensional time series
Spatial Power	$\phi^{\|s_{ij} - s_{ik}\|}$	Exponential decay	Distance-based structure	Used when correlation depends on spatial distance	Spatial statistics, environmental data
Matérn Covariance	Function of $\|t_{ij} - t_{ik}\|^{\nu}$	Flexible decay	Spatially continuous	Generalization of Gaussian and exponential structures	Geostatistics, spatiotemporal models

---

Choosing the Right Covariance Structure

Scenario	Recommended Structure
Repeated measures with equal correlation	Compound Symmetry
Longitudinal data with time dependence	AR(1), Toeplitz
Continuous-time process	Exponential, Gaussian
Spatial correlation	Spatial Power, Matérn
Irregularly spaced time points	Toeplitz, Unstructured
High-dimensional data	Banded, AR(1)

8.2 Estimation in Linear Mixed Models

The general Linear Mixed Model is:

$$\mathbf{Y}_i = \mathbf{X}_i\beta + \mathbf{Z}_i\mathbf{b}_i + \epsilon_i,$$

where:

- β: Fixed effects (parameters shared across subjects).
- $\mathbf{b}_i$: Random effects (subject-specific deviations).
- $\mathbf{X}_i$: Design matrix for fixed effects.
- $\mathbf{Z}_i$: Design matrix for random effects.
- $\mathbf{D}$: Covariance matrix of the random effects.
- $_i$: Covariance matrix of the residual errors.

Since β, $\mathbf{b}_i$, $\mathbf{D}$, and $_i$ are **unknown**, they must be **estimated** from data.

- $\beta, \mathbf{D}, \ _i$ are **fixed parameters** $\rightarrow$ must be **estimated**.
- $\mathbf{b}_i$ is a **random variable** $\rightarrow$ must be **predicted** (not estimated). In other words, random thing/variable can't be estimated.

Thus, we define:

- **Estimator of** β: $\hat{\beta}$ (fixed effect estimation).
- **Predictor of** $\mathbf{b}_i$: $\hat{\mathbf{b}}_i$ (random effect prediction).

Then:

- The **population-level estimate** of $\mathbf{Y}_i$ is:

$$\hat{\mathbf{Y}}_i = \mathbf{X}_i\hat{\beta}.$$

- The **subject-specific prediction** is:

$$\hat{\mathbf{Y}}_i = \mathbf{X}_i\hat{\beta} + \mathbf{Z}_i\hat{\mathbf{b}}_i.$$

For all N subjects, we stack the equations into the Mixed Model Equations (Henderson, 1975):

$$\mathbf{Y} = \mathbf{X}\beta + \mathbf{Z}\mathbf{b} + \epsilon.$$

and

$$\mathbf{Y} \sim N(\mathbf{X} \ , \mathbf{Z}\mathbf{B}\mathbf{Z}' + \ )$$

where:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_N \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_N \end{bmatrix}, \quad = \begin{bmatrix} {}_1 \\ \vdots \\ {}_N \end{bmatrix}.$$

The covariance structure is:

$$\mathrm{Cov}(\mathbf{b}) = \mathbf{B}, \quad \mathrm{Cov}(\epsilon) = \ , \quad \mathrm{Cov}(\mathbf{b}, \epsilon) = 0.$$

Expanding $\mathbf{Z}$ and $\mathbf{B}$ as block diagonal matrices:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_1 & 0 & \dots & 0 \\ 0 & \mathbf{Z}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbf{Z}_N \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{D} & 0 & \dots & 0 \\ 0 & \mathbf{D} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbf{D} \end{bmatrix}.$$

The best linear unbiased estimator (BLUE) for β and the best linear unbiased predictor (BLUP) for $\mathbf{b}$ are obtained by solving (Henderson, 1975):

$$\begin{bmatrix} \hat{\beta} \\ \hat{\mathbf{b}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\ {}^{-1}\mathbf{X} & \mathbf{X}'\ {}^{-1}\mathbf{Z} \\ \mathbf{Z}'\ {}^{-1}\mathbf{X} & \mathbf{Z}'\ {}^{-1}\mathbf{Z} + \mathbf{B}^{-1} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{X}'\ {}^{-1}\mathbf{Y} \\ \mathbf{Z}'\ {}^{-1}\mathbf{Y} \end{bmatrix}.$$

where:

- $\hat{\beta}$ is the Generalized Least Squares estimator for β.
- $\hat{\mathbf{b}}$ is the BLUP for $\mathbf{b}$.

If we define:

$$\mathbf{V} = \mathbf{Z}\mathbf{B}\mathbf{Z}' + \ .$$

Then, the solutions to the **Mixed Model Equations** are:

$$\hat{\beta} = (\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{Y},$$
$$\hat{\mathbf{b}} = \mathbf{B}\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{Y} - \mathbf{X}\ ).$$

where:

- $\hat{\beta}$ is obtained using Generalized Least Squares.
- $\hat{\mathbf{b}}$ is a Weighted Least Squares predictor, where weights come from $\mathbf{B}$ and $\mathbf{V}$.

Properties of the Estimators

For $\hat{\beta}$:

$$E(\hat{\beta}) = \beta, \quad \text{Var}(\hat{\beta}) = (\mathbf{X'V^{-1}X})^{-1}.$$

For $\hat{\mathbf{b}}$:

$$E(\hat{\mathbf{b}}) = 0.$$

The **variance of the prediction error** (Mean Squared Prediction Error, MSPE) is:

$$\text{Var}(\hat{\mathbf{b}} - \mathbf{b}) = \mathbf{B} - \mathbf{BZ'V^{-1}ZB} + \mathbf{BZ'V^{-1}X}(\mathbf{X'V^{-1}X})^{-1}\mathbf{X'V^{-1}B}.$$

Key Insight:
Mean Squared Prediction Error is more meaningful than $\text{Var}(\hat{\mathbf{b}})$ alone, since it accounts for both variance and bias in prediction.

8.2.1 Interpretation of the Mixed Model Equations

The system:

$$\begin{bmatrix} \mathbf{X}'^{\,-1}\mathbf{X} & \mathbf{X}'^{\,-1}\mathbf{Z} \\ \mathbf{Z}'^{\,-1}\mathbf{X} & \mathbf{Z}'^{\,-1}\mathbf{Z} + \mathbf{B}^{-1} \end{bmatrix} \begin{bmatrix} \beta \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'^{\,-1}\mathbf{Y} \\ \mathbf{Z}'^{\,-1}\mathbf{Y} \end{bmatrix}.$$

can be understood as:

1. Fixed Effects Estimation ($\hat{\beta}$)
   - Uses Generalized Least Squares.
   - Adjusted for both random effects and correlated errors.
2. Random Effects Prediction ($\hat{\mathbf{b}}$)
   - Computed using the BLUP formula.
   - Shrinks subject-specific estimates toward the population mean.

Component	**Equation**	**Interpretation**
Fixed effects ($\hat{\beta}$)	$(\mathbf{X'V^{-1}X})^{-1}\mathbf{X'V^{-1}Y}$	Generalized Least Squares estimator
Random effects ($\hat{\mathbf{b}}$)	$\mathbf{BZ'V^{-1}}(\mathbf{Y} - \mathbf{X}\ )$	Best Linear Unbiased Predictor (BLUP)

Component	Equation	Interpretation
Variance of $\hat{\beta}$	$(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}$	Accounts for uncertainty in fixed effect estimates
Variance of prediction error	$\mathbf{B} - \mathbf{B}\mathbf{Z}'\mathbf{V}^{-1}\mathbf{Z}\mathbf{B} + \mathbf{B}\mathbf{Z}'\mathbf{V}^{-1}\mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{B}$	Includes both variance and bias in prediction

---

8.2.2 Derivation of the Mixed Model Equations

To derive the **Mixed Model Equations**, consider:

$$= \mathbf{Y} - \mathbf{X} \ - \mathbf{Z}\mathbf{b}.$$

Define:

- $T = \sum_{i=1}^{N} n_i \rightarrow$ **Total number of observations**.
- $Nq \rightarrow$ **Total number of random effects**.

The **joint distribution** of $(\mathbf{b}, \ )$ is:

$$f(\mathbf{b}, \epsilon) = \frac{1}{(2\pi)^{(T+Nq)/2}} \begin{vmatrix} \mathbf{B} & 0 \\ 0 & \end{vmatrix}^{-1/2}$$
$$\exp\left(-\frac{1}{2}\begin{bmatrix} \mathbf{b} \\ \mathbf{Y} - \mathbf{X} \ - \mathbf{Z}\mathbf{b} \end{bmatrix}' \begin{bmatrix} \mathbf{B} & 0 \\ 0 & \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{b} \\ \mathbf{Y} - \mathbf{X} \ - \mathbf{Z}\mathbf{b} \end{bmatrix}\right).$$

Maximizing $f(\mathbf{b}, \epsilon)$ with respect to $\mathbf{b}$ and β requires **minimization** of:

$$Q = \begin{bmatrix} \mathbf{b} \\ \mathbf{Y} - \mathbf{X} \ - \mathbf{Z}\mathbf{b} \end{bmatrix}' \begin{bmatrix} \mathbf{B} & 0 \\ 0 & \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{b} \\ \mathbf{Y} - \mathbf{X} \ - \mathbf{Z}\mathbf{b} \end{bmatrix}$$
$$= \mathbf{b}'\mathbf{B}^{-1}\mathbf{b} + (\mathbf{Y} - \mathbf{X} \ - \mathbf{Z}\mathbf{b})' \ ^{-1}(\mathbf{Y} - \mathbf{X} \ - \mathbf{Z}\mathbf{b}).$$

Setting the derivatives of Q with respect to $\mathbf{b}$ and to zero leads to the system of equations:

$$\mathbf{X}' \ ^{-1}\mathbf{X} \ + \mathbf{X}' \ ^{-1}\mathbf{Z}\mathbf{b} = \mathbf{X}' \ ^{-1}\mathbf{Y}$$
$$(\mathbf{Z}' \ ^{-1}\mathbf{Z} + \mathbf{B}^{-1})\mathbf{b} + \mathbf{Z}' \ ^{-1}\mathbf{X} \ = \mathbf{Z}' \ ^{-1}\mathbf{Y}$$

Rearranging

$$\begin{bmatrix} \mathbf{X}' \ ^{-1}\mathbf{X} & \mathbf{X}' \ ^{-1}\mathbf{Z} \\ \mathbf{Z}' \ ^{-1}\mathbf{X} & \mathbf{Z}' \ ^{-1}\mathbf{Z} + \mathbf{B}^{-1} \end{bmatrix} \begin{bmatrix} \beta \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} \mathbf{X}' \ ^{-1}\mathbf{Y} \\ \mathbf{Z}' \ ^{-1}\mathbf{Y} \end{bmatrix}$$

Thus, the solution to the mixed model equations give:

$$\begin{bmatrix} \hat{\beta} \\ \hat{\mathbf{b}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'^{\,-1}\mathbf{X} & \mathbf{X}'^{\,-1}\mathbf{Z} \\ \mathbf{Z}'^{\,-1}\mathbf{X} & \mathbf{Z}'^{\,-1}\mathbf{Z} + \mathbf{B}^{-1} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{X}'^{\,-1}\mathbf{Y} \\ \mathbf{Z}'^{\,-1}\mathbf{Y} \end{bmatrix}$$

8.2.3 Bayesian Interpretation of Linear Mixed Models

In a **Bayesian framework**, the posterior distribution of the **random effects** $\mathbf{b}$ given the observed data $\mathbf{Y}$ is derived using **Bayes' theorem**:

$$f(\mathbf{b}|\mathbf{Y}) = \frac{f(\mathbf{Y}|\mathbf{b})f(\mathbf{b})}{\int f(\mathbf{Y}|\mathbf{b})f(\mathbf{b})d\mathbf{b}}.$$

where:

- $f(\mathbf{Y}|\mathbf{b})$ is the likelihood function, describing how the data are generated given the random effects.
- $f(\mathbf{b})$ is the prior distribution of the random effects.
- The denominator $\int f(\mathbf{Y}|\mathbf{b})f(\mathbf{b})d\mathbf{b}$ is the normalizing constant that ensures the posterior integrates to 1.
- $f(\mathbf{b}|\mathbf{Y})$ is the posterior distribution, which updates our belief about $\mathbf{b}$ given the observed data $\mathbf{Y}$.

In the Linear Mixed Model, we assume:

$$\begin{aligned} \mathbf{Y}|\mathbf{b} &\sim N(\mathbf{X}\ + \mathbf{Zb},\ ), \\ \mathbf{b} &\sim N(\mathbf{0}, \mathbf{B}). \end{aligned}$$

This means:

- Likelihood: Given $\mathbf{b}$, the data $\mathbf{Y}$ follows a multivariate normal distribution with mean $\mathbf{X}\ + \mathbf{Zb}$ and covariance .
- Prior for $\mathbf{b}$: The random effects are assumed to follow a multivariate normal distribution with mean 0 and covariance $\mathbf{B}$.

By applying **Bayes' theorem**, the posterior distribution of $\mathbf{b}$ given $\mathbf{Y}$ is:

$$\mathbf{b}|\mathbf{Y} \sim N(\mathbf{BZ}'\mathbf{V}^{-1}(\mathbf{Y} - \mathbf{X}\ ), (\mathbf{Z}'^{\,-1}\mathbf{Z} + \mathbf{B}^{-1})^{-1}).$$

where:

- Mean: $\mathbf{BZ}'\mathbf{V}^{-1}(\mathbf{Y} - \mathbf{X}\ )$
  - This is the BLUP.

  - It represents the expected value of $\mathbf{b}$ given $\mathbf{Y}$ under squared-error loss.
- Covariance: $(\mathbf{Z}'\ ^{-1}\mathbf{Z} + \mathbf{B}^{-1})^{-1}$
  - This posterior variance accounts for both prior uncertainty ($\mathbf{B}$) and data uncertainty ( ).

Thus, the Bayesian posterior mean of $\mathbf{b}$ coincides with the BLUP predictor:

$$E(\mathbf{b}|\mathbf{Y}) = \mathbf{B}\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{Y} - \mathbf{X}\ ).$$

Interpretation of the Posterior Distribution

1. Posterior Mean as a Shrinkage Estimator (BLUP)
   - The expectation $E(\mathbf{b}|\mathbf{Y})$ shrinks individual estimates toward the population mean.
   - Subjects with less data or more variability will have estimates closer to zero.
   - This is similar to Ridge Regression in penalized estimation.
2. Posterior Variance Quantifies Uncertainty
   - The matrix $(\mathbf{Z}'\ ^{-1}\mathbf{Z} + \mathbf{B}^{-1})^{-1}$ captures the remaining uncertainty in $\mathbf{b}$ after seeing $\mathbf{Y}$.
   - If $\mathbf{Z}'\ ^{-1}\mathbf{Z}$ is large, the data provide strong information about $\mathbf{b}$, reducing posterior variance.
   - If $\mathbf{B}^{-1}$ dominates, prior information heavily influences estimates.
3. Connection to Bayesian Inference
   - The random effects $\mathbf{b}$ follow a Gaussian posterior due to conjugacy.
   - This is analogous to Bayesian hierarchical models, where random effects are latent variables estimated from data.

Step	**Equation**	**Interpretation**
Likelihood	$\mathbf{Y}\|\mathbf{b} \sim N(\mathbf{X}\ + \mathbf{Z}\mathbf{b},\ )$	Data given random effects
Prior	$\mathbf{b} \sim N(\mathbf{0}, \mathbf{B})$	Random effects distribution
Posterior	$\mathbf{b}\|\mathbf{Y} \sim N(\mathbf{B}\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{Y} - \mathbf{X}\ ), (\mathbf{Z}'\ ^{-1}\mathbf{Z} + \mathbf{B}^{-1})^{-1})$	Updated belief about $\mathbf{b}$
Posterior Mean (BLUP)	$E(\mathbf{b}\|\mathbf{Y}) = \mathbf{B}\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{Y} - \mathbf{X}\ )$	Best predictor (squared error loss)
Posterior Variance	$(\mathbf{Z}'\ ^{-1}\mathbf{Z} + \mathbf{B}^{-1})^{-1}$	Uncertainty in predictions

8.2.4 Estimating the Variance-Covariance Matrix

If we have an estimate $\tilde{\mathbf{V}}$ for $\mathbf{V}$, we can estimate the fixed and random effects as:

$$\hat{\beta} = (\mathbf{X}'\tilde{\mathbf{V}}^{-1}\mathbf{X})^{-1}\mathbf{X}'\tilde{\mathbf{V}}^{-1}\mathbf{Y},$$
$$\hat{\mathbf{b}} = \mathbf{B}\mathbf{Z}'\tilde{\mathbf{V}}^{-1}(\mathbf{Y} - \mathbf{X}\ ).$$

where:

- $\hat{\beta}$ is the estimated fixed effects.
- $\hat{\mathbf{b}}$ is the Empirical Best Linear Unbiased Predictor (EBLUP), also called the Empirical Bayes estimate of $\mathbf{b}$.

Properties of $\hat{\beta}$ and Variance Estimation

- **Consistency**: $\hat{\text{Var}}(\hat{\beta})$ is a consistent estimator of $\text{Var}(\hat{\beta})$ if $\tilde{\mathbf{V}}$ is a consistent estimator of $\mathbf{V}$.
- **Bias Issue**: $\hat{\text{Var}}(\hat{\beta})$ is biased because it does not account for the uncertainty in estimating $\mathbf{V}$.
- **Implication**: This means that $\hat{\text{Var}}(\hat{\beta})$ underestimates the true variability.

To estimate $\mathbf{V}$, several approaches can be used:

- Maximum Likelihood Estimation (MLE)
- Restricted Maximum Likelihood (REML)
- Estimated Generalized Least Squares (EGLS)
- Bayesian Hierarchical Models (BHM)

8.2.4.1 Maximum Likelihood Estimation

MLE finds parameter estimates by maximizing the likelihood function.

Define a parameter vector θ that includes all unknown variance components in and $\mathbf{B}$. Then, we assume:

$$\mathbf{Y} \sim N(\mathbf{X}\ , \mathbf{V}(\theta)).$$

The log-likelihood function (ignoring constant terms) is:

$$-2\log L(\mathbf{y}; \theta, \beta) = \log|\mathbf{V}(\theta)| + (\mathbf{Y} - \mathbf{X}\ )'\mathbf{V}(\theta)^{-1}(\mathbf{Y} - \mathbf{X}\ ).$$

Steps for MLE Estimation

1. Estimate $\hat{\beta}$, assuming θ is known:

$$\hat{\beta}_{MLE} = (\mathbf{X}'\mathbf{V}(\ )^{-\mathbf{1}}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}(\ )^{-\mathbf{1}}\mathbf{Y}.$$

2. Obtain $\hat{\theta}_{MLE}$ by maximizing the log-likelihood:

$$\hat{\theta}_{MLE} = \arg\max_{\theta} -2\log L(\mathbf{y};\theta,\beta).$$

3. Substitute $\hat{\theta}_{MLE}$ to get updated estimates:

$$\hat{\beta}_{MLE} = (\mathbf{X}'\mathbf{V}(\hat{\ }_{\mathbf{MLE}})^{-\mathbf{1}}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}(\hat{\ }_{\mathbf{MLE}})^{-\mathbf{1}}\mathbf{Y}.$$

4. Predict random effects:

$$\hat{\mathbf{b}}_{MLE} = \mathbf{B}(\hat{\theta}_{MLE})\mathbf{Z}'\mathbf{V}(\hat{\theta}_{MLE})^{-1}(\mathbf{Y} - \mathbf{X}\hat{\ }_{\mathbf{MLE}}).$$

Key Observations about MLE

- MLE tends to underestimate θ because it does not account for the estimation of fixed effects.
- Bias in variance estimates can be corrected using REML.

8.2.4.2 Restricted Maximum Likelihood

Restricted Maximum Likelihood (REML) is an estimation method that improves upon Maximum Likelihood Estimation by accounting for the loss of degrees of freedom due to the estimation of fixed effects.

Unlike MLE, which estimates both fixed effects (β) and variance components (θ) simultaneously, REML focuses on estimating variance components by considering linear combinations of the data that are independent of the fixed effects.

Consider the Linear Mixed Model:

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{Z}\mathbf{b} + \epsilon,$$

where:

- $\mathbf{y}$: Response vector of length N
- $\mathbf{X}$: Design matrix for fixed effects ($N \times p$)
- β: Fixed effects parameter vector ($p \times 1$)
- $\mathbf{Z}$: Design matrix for random effects
- $\mathbf{b} \sim N(\mathbf{0}, \mathbf{D})$: Random effects
- $\epsilon \sim N(\mathbf{0},\ )$: Residual errors

The marginal distribution of $\mathbf{y}$ is:

$$\mathbf{y} \sim N(\mathbf{X}\beta, \mathbf{V}(\theta)),$$

where:

$$\mathbf{V}(\theta) = \mathbf{ZDZ}' + \ .$$

To eliminate dependence on β, consider linear transformations of $\mathbf{y}$ that are orthogonal to the fixed effects.

Let $\mathbf{K}$ be a full-rank contrast matrix of size $N \times (N - p)$ such that:

$$\mathbf{K}'\mathbf{X} = 0.$$

Then, we consider the transformed data:

$$\mathbf{K}'\mathbf{y} \sim N(\mathbf{0}, \mathbf{K}'\mathbf{V}(\theta)\mathbf{K}).$$

- This transformation removes β from the likelihood, focusing solely on the variance components θ.
- Importantly, the choice of $\mathbf{K}$ does not affect the final REML estimates.

The REML log-likelihood is:

$$-2 \log L_{REML}(\theta) = \log |\mathbf{K}'\mathbf{V}(\theta)\mathbf{K}| + \mathbf{y}'\mathbf{K}(\mathbf{K}'\mathbf{V}(\theta)\mathbf{K})^{-1}\mathbf{K}'\mathbf{y}.$$

An equivalent form of the REML log-likelihood, avoiding explicit use of $\mathbf{K}$, is:

$$-2 \log L_{REML}(\theta) = \log |\mathbf{V}(\theta)| + \log |\mathbf{X}'\mathbf{V}(\theta)^{-1}\mathbf{X}| + (\mathbf{y} - \mathbf{X}\hat{\beta})'\mathbf{V}(\theta)^{-1}(\mathbf{y} - \mathbf{X}\hat{\beta}),$$

where:

$$\hat{\beta} = (\mathbf{X}'\mathbf{V}(\theta)^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}(\theta)^{-1}\mathbf{y}.$$

This form highlights how REML adjusts for the estimation of fixed effects via the second term $\log |\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}|$.

Steps for REML Estimation

1. Transform the data using $\mathbf{K}'\mathbf{y}$ to remove β from the likelihood.
2. Maximize the restricted likelihood to estimate $\hat{\theta}_{REML}$.

3. Estimate fixed effects using:

$$\hat{\beta}_{REML} = (\mathbf{X}'\mathbf{V}(\hat{\theta}_{REML})^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}(\hat{\theta}_{REML})^{-1}\mathbf{y}.$$

Properties of REML

1. **Unbiased Variance Component Estimates**: REML produces unbiased estimates of variance components by accounting for the degrees of freedom used to estimate fixed effects.
2. **Invariance to Fixed Effects**: The restricted likelihood is constructed to be independent of the fixed effects β.
3. **Asymptotic Normality**: REML estimates are consistent and asymptotically normal under standard regularity conditions.
4. **Efficiency**: While REML estimates variance components efficiently, it does not maximize the joint likelihood of all parameters, so β estimates are slightly less efficient compared to MLE.

Table 8.6: Comparison of REML and MLE

Criterion	MLE	REML
Approach	Maximizes full likelihood	Maximizes likelihood of contrasts (removes β)
Estimates Fixed Effects?	Yes	No (focuses on variance components)
Bias in Variance Estimates	Biased (underestimates variance components)	Unbiased (corrects for loss of degrees of freedom)
Effect of Changing $\mathbf{X}$	Affects variance estimates	No effect on variance estimates
Consistency	Yes	Yes
Asymptotic Normality	Yes	Yes
Efficiency	Efficient under normality	More efficient for variance components
Model Comparison (AIC/BIC)	Suitable for comparing different fixed-effect models	Not suitable (penalizes fixed effects differently)
Performance in Small Samples	Sensitive to small sample bias	More robust to small sample bias
Handling Outliers	More sensitive	Less sensitive
Equivalent to ANOVA?	No	Yes, in balanced designs

8.2.4.3 Estimated Generalized Least Squares

MLE and REML rely on the Gaussian assumption, which may not always hold. EGLS provides an alternative by relying only on the first two moments (mean and variance).

The LMM framework is:

$$\mathbf{Y}_i = \mathbf{X}_i\beta + \mathbf{Z}_i\mathbf{b}_i + \epsilon_i.$$

where:

- Random effects: $\mathbf{b}_i \sim N(\mathbf{0}, \mathbf{D})$.
- Residual errors: $\epsilon_i \sim N(\mathbf{0},\ _{\mathbf{i}})$.
- Independence assumption: $\text{Cov}(\epsilon_i, \mathbf{b}_i) = 0$.

Thus, the first two moments are:

$$E(\mathbf{Y}_i) = \mathbf{X}_i\beta, \quad \text{Var}(\mathbf{Y}_i) = \mathbf{V}_i.$$

The EGLS estimator is:

$$\hat{\beta}_{GLS} = \left\{\sum_{i=1}^{n} \mathbf{X_i'V_i}(\ )^{-\mathbf{1}}\mathbf{X_i}\right\}^{-1} \sum_{i=1}^{n} \mathbf{X_i'V_i}(\ )^{-\mathbf{1}}\mathbf{Y_i}.$$

Writing in matrix form:

$$\hat{\beta}_{GLS} = \left\{\mathbf{X'V}(\ )^{-\mathbf{1}}\mathbf{X}\right\}^{-1} \mathbf{X'V}(\ )^{-\mathbf{1}}\mathbf{Y}.$$

Since $\mathbf{V}(\theta)$ is unknown, we estimate it as $\hat{\mathbf{V}}$, leading to the EGLS estimator:

$$\hat{\beta}_{EGLS} = \left\{\mathbf{X'\hat{V}^{-1}X}\right\}^{-1} \mathbf{X'\hat{V}^{-1}Y}.$$

Key Insights about EGLS

- Computational Simplicity:
  - EGLS does not require iterative maximization of a likelihood function, making it computationally attractive.
- Same Form as MLE/REML:
  - The fixed effects estimators for MLE, REML, and EGLS have the same form, differing only in how $\mathbf{V}$ is estimated.
- Robust to Non-Gaussian Data:

  - Since it only depends on first and second moments, it can handle cases where MLE and REML struggle with non-normality.

When to Use EGLS?

- When the normality assumption for MLE/REML is questionable.
- When $\mathbf{V}$ can be estimated efficiently without requiring complex optimization.
- In non-iterative approaches, where computational simplicity is a priority.

8.2.4.4 Bayesian Hierarchical Models

Bayesian methods offer a fully probabilistic framework to estimate $\mathbf{V}$ by incorporating prior distributions.

The joint distribution can be decomposed hierarchically:

$$f(A, B, C) = f(A|B, C)f(B|C)f(C).$$

Applying this to LMMs:

$$\begin{aligned} f(\mathbf{Y}, \ , \mathbf{b}, \ ) &= f(\mathbf{Y}| \ , \mathbf{b}, \ )f(\mathbf{b}| \ , \ )f(\ | \ )f(\ ) \\ &= f(\mathbf{Y}| \ , \mathbf{b}, \ )f(\mathbf{b}| \ )f(\ )f(\ ). \end{aligned}$$

where:

- The first equality follows from probability decomposition.
- The second equality assumes conditional independence, meaning:
  - Given θ, no additional information about $\mathbf{b}$ is obtained from knowing β.

Using Bayes' theorem, the posterior distribution is:

$$f(\ , \mathbf{b}, \ |\mathbf{Y}) \propto f(\mathbf{Y}| \ , \mathbf{b}, \ )f(\mathbf{b}| \ )f(\ )f(\ ).$$

where:

$$\begin{aligned} \mathbf{Y}| \ , \mathbf{b}, \ &\sim N(\mathbf{X} \ + \mathbf{Zb}, \ (\ )), \\ \mathbf{b}| \ &\sim N(\mathbf{0}, \mathbf{B}(\ )). \end{aligned}$$

To complete the Bayesian model, we specify prior distributions:

- $f(\beta)$: Prior on fixed effects.
- $f(\theta)$: Prior on variance components.

Since analytical solutions are generally unavailable, we use Markov Chain Monte Carlo (MCMC) to sample from the posterior:

- Gibbs sampling (if conjugate priors are used).
- Hamiltonian Monte Carlo (HMC) (for complex models).

Advantages

1. Accounts for Parameter Uncertainty
   - Unlike MLE/REML, Bayesian methods propagate uncertainty in variance component estimation.
2. Flexible Model Specification
   - Can incorporate prior knowledge via informative priors.
   - Extends naturally beyond Gaussian assumptions (e.g., Student-t distributions for heavy-tailed errors).
3. Robustness in Small Samples
   - Bayesian methods can stabilize variance estimation in small datasets where MLE/REML are unreliable.

Challenges

1. Computational Complexity
   - Requires MCMC algorithms, which can be computationally expensive.
2. Convergence Issues
   - MCMC chains must be checked for convergence (e.g., using R-hat diagnostic).
3. Choice of Priors
   - Poorly chosen priors can bias estimates or slow down convergence.

Comparison of Estimation Methods for $\mathbf{V}$

Method	Assumptions	Computational Cost	Handles Non-Normality?	Best Use Case
MLE	Gaussian errors	High (iterative)	No	Model selection (AIC/BIC)
REML	Gaussian errors	High (iterative)	No	Variance estimation
EGLS	First two moments	Low (non-iterative)	Yes	Large-scale models with correlated errors
Bayesian (BHM)	Probabilistic	Very High (MCMC)	Yes	Small samples, prior information available

8.3 Inference in Linear Mixed Models

8.3.1 Inference for Fixed Effects (β)

The goal is to test hypotheses about the fixed effects parameters β using various statistical tests:

- Wald Test
- F-Test
- Likelihood Ratio Test

8.3.1.1 Wald Test

The Wald test assesses whether certain linear combinations of fixed effects are equal to specified values.

Given:

$$\hat{\beta}(\theta) = \left(\mathbf{X}'\mathbf{V}^{-1}(\theta)\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}(\theta)\mathbf{Y},$$

and its variance:

$$\mathrm{Var}(\hat{\beta}(\theta)) = \left(\mathbf{X}'\mathbf{V}^{-1}(\theta)\mathbf{X}\right)^{-1}.$$

In practice, we substitute $\hat{\theta}$ (the estimate of θ) to obtain:

- Hypotheses:

$$H_0 : \mathbf{A} \ = \mathbf{d}$$

  where:

  - $\mathbf{A}$ is a contrast matrix specifying linear combinations of β.
  - $\mathbf{d}$ is a constant vector representing the null hypothesis values.

- Wald Test Statistic:

$$W = (\mathbf{A}\hat{\beta} - \mathbf{d})' \left[\mathbf{A}\left(\mathbf{X}'\hat{\mathbf{V}}^{-1}\mathbf{X}\right)^{-1}\mathbf{A}'\right]^{-1}(\mathbf{A}\hat{\beta} - \mathbf{d}).$$

- Distribution under H_0:

$$W \sim \chi^2_{\mathrm{rank}(\mathbf{A})}.$$

Caution with Wald Test:

- Underestimation of Variance:
  The Wald test ignores the variability from estimating $\hat{\theta}$, leading to underestimated standard errors and potentially inflated Type I error rates.
- Small Sample Issues:
  Less reliable in small samples or when variance components are near boundary values (e.g., variances close to zero).

8.3.1.2 F-Test

An alternative to the Wald test, the F-test adjusts for the estimation of σ^2 and provides better performance in small samples.

Assume:

$$\text{Var}(\mathbf{Y}) = \sigma^2 \mathbf{V}(\theta).$$

The F-statistic is:

$$F^* = \frac{(\mathbf{A}\hat{\beta} - \mathbf{d})' \left[\mathbf{A}\left(\mathbf{X}'\hat{\mathbf{V}}^{-1}\mathbf{X}\right)^{-1}\mathbf{A}'\right]^{-1} (\mathbf{A}\hat{\beta} - \mathbf{d})}{\hat{\sigma}^2 \,\text{rank}(\mathbf{A})}.$$

- Distribution under H_0:

$$F^* \sim F_{\text{rank}(\mathbf{A}),\, \text{df}_{\text{denominator}}}.$$

- Approximating Denominator Degrees of Freedom:
  - Satterthwaite approximation
  - Kenward-Roger approximation (provides bias-corrected standard errors)

F-Test Advantages:

- More accurate in small samples compared to the Wald test.
- Adjusts for variance estimation, reducing bias in hypothesis testing.

Wald Test vs. F-Test:

Criterion	Wald Test	F-Test
Small Sample Performance	Poor (can inflate Type I error)	Better control of Type I error
Variance Estimation	Ignores variability in $\hat{\theta}$	Adjusts using $\hat{\sigma}^2$

Criterion	Wald Test	F-Test
Reduction to t-test	Yes (for single β)	Yes (when rank($\mathbf{A}$) = 1)

8.3.1.3 Likelihood Ratio Test

The Likelihood Ratio Test (LRT) compares the fit of nested models:

- Null Hypothesis:

$$H_0 : \beta \in \Theta_{\beta,0}$$

  where $\Theta_{\beta,0}$ is a subset of the full parameter space Θ_β.

- Test Statistic:

$$-2\log\lambda = -2\log\left(\frac{\hat{L}_{ML,0}}{\hat{L}_{ML}}\right),$$

  where:
  - $\hat{L}_{ML,0}$ = Maximized likelihood under H_0 (restricted model)
  - $\hat{L}_{ML}$ = Maximized likelihood under the alternative (full model)

- Distribution under H_0:

$$-2\log\lambda \sim \chi^2_{df}$$

  where $df = \dim(\Theta_\beta) - \dim(\Theta_{\beta,0})$ (the difference in the number of parameters).

Important Notes:

- LRT is applicable only for ML estimates (not REML) when comparing models with different fixed effects.
- REML-based LRT can be used for comparing models that differ in random effects (variance components), but not fixed effects.

8.3.2 Inference for Variance Components (θ)

For ML and REML estimators:

$$\hat{\theta} \sim N(\theta, I(\theta)^{-1}),$$

where $I(\theta)$ is the Fisher Information Matrix.

This normal approximation holds well for large samples, enabling Wald-type tests and confidence intervals.

8.3.2.1 Wald Test for Variance Components

The Wald test for variance components follows the same structure as for fixed effects:

- Test Statistic:

$$W = \frac{(\hat{\theta} - \theta_0)^2}{\widehat{\mathrm{Var}}(\hat{\theta})}.$$

- Distribution under H_0:

$$W \sim \chi^2_1.$$

Limitations of Wald Test for Variance Components:

- Boundary Issues: The normal approximation fails when the true variance component is near zero (boundary of the parameter space).
- Less reliable for variance parameters than for covariance parameters.

8.3.2.2 Likelihood Ratio Test for Variance Components

LRT can also be applied to variance components:

- Test Statistic:

$$-2\log\lambda = -2\log\left(\frac{\hat{L}_{REML,0}}{\hat{L}_{REML}}\right).$$

- Distribution under H_0:
  - Not always χ^2-distributed when variance components are on the boundary (e.g., testing if $\sigma^2 = 0$).
  - May require mixture distributions or adjusted critical values.

Test	Best For	Strengths	Limitations
Wald Test	Fixed effects (β)	Simple, widely used	Underestimates variance, biased in small samples

Test	Best For	Strengths	Limitations
F-Test	Fixed effects (β)	Better in small samples, adjusts df	Requires approximation for degrees of freedom
LRT (ML)	Fixed effects, nested models	Powerful, widely used	Not valid for REML with fixed effects
LRT (REML)	Variance components	Robust for random effects	Boundary issues when variances are near zero
Wald (Variance)	Variance components (θ)	Simple extension of Wald test	Fails near parameter space boundaries

8.4 Information Criteria for Model Selection

Information Criteria are statistical tools used to compare competing models by balancing model fit (likelihood) with model complexity (number of parameters). They help in identifying the most parsimonious model that adequately explains the data without overfitting.

The three most commonly used criteria are:

- Akaike Information Criterion (AIC)
- Corrected Akaike Information Criterion (AICc)
- Bayesian Information Criterion (BIC)

8.4.1 Akaike Information Criterion

The Akaike Information Criterion is derived from the Kullback-Leibler divergence, which measures the difference between the true data-generating process and the fitted model.

AIC Formula:

$$AIC = -2\,l(\hat{\theta}, \hat{\beta}) + 2q$$

where:

- $l(\hat{\theta}, \hat{\beta})$: The maximized log-likelihood of the model, evaluated at the estimates $\hat{\theta}$ (variance components) and $\hat{\beta}$ (fixed effects).
- q: The effective number of parameters, including:
  - The number of fixed effects.
  - The number of variance-covariance parameters (random effects).
  - Excludes parameters constrained to boundary values (e.g., variances estimated as zero).

Key Points About AIC

- Model Selection Rule:
  - Lower AIC indicates a better model.
  - Occasionally, software may report AIC as $l - q$, in which case higher AIC is better (rare).
- Comparing Random Effects Models:
  - Not recommended when comparing models with different random effects because it's difficult to accurately count the effective number of parameters.
- Sample Size Considerations:
  - Requires large sample sizes for reliable comparisons.
  - In small samples, AIC tends to favor more complex models due to insufficient penalty for model complexity.
- Potential Bias:
  - Can be negatively biased (i.e., favoring overly complex models) when the sample size is small relative to the number of parameters.

When to Use AIC:

- Comparing models with the same random effects structure but different fixed effects.
- Selecting covariance structures in mixed models when the sample size is large.

8.4.2 Corrected AIC

The Corrected AIC (AICc) addresses the bias in AIC for small sample sizes. It was developed by (Hurvich and Tsai, 1989).

AICc Formula:

$$AICc = AIC + \frac{2q(q+1)}{n-q-1}$$

where:

- n: The sample size.
- q: The number of estimated parameters.

Key Points About AICc

- Small Sample Correction:
  - Provides a stronger penalty for model complexity when the sample size is small.
- Applicability:
  - Valid when comparing models with fixed covariance structures.

  - Not recommended for models with general covariance structures due to difficulties in bias correction.
- Model Selection Rule:
  - Lower AICc indicates a better model.

When to Use AICc:

- Small sample sizes (n/q ratio is low).
- Models with fixed random effects or simple covariance structures.

8.4.3 Bayesian Information Criterion

The Bayesian Information Criterion is derived from a Bayesian framework and incorporates a stronger penalty for model complexity compared to AIC.

BIC Formula

$$BIC = -2\,l(\hat{\theta}, \hat{\beta}) + q\log(n)$$

where:

- n: The number of observations.
- q: The number of effective parameters.
- $l(\hat{\theta}, \hat{\beta})$: The maximized log-likelihood.

Key Points About BIC

- Model Selection Rule:
  - Lower BIC indicates a better model.
- Stronger Penalty:
  - The penalty term $q\log(n)$ grows with sample size, leading BIC to favor simpler models more than AIC.
- Applicability to MLE and REML:
  - BIC can be used with both MLE and REML, but:
    * Use MLE when comparing models with different fixed effects.
    * Use REML when comparing models with different random effects (same fixed effects).
- Consistency:
  - BIC is consistent, meaning that as the sample size increases, it will select the true model with probability 1 (if the true model is among the candidates).

When to Use BIC:

- Large sample sizes where model simplicity is prioritized.
- Model selection for hypothesis testing (due to its connection to Bayesian inference).

Comparison of AIC, AICc, and BIC

Criterion	Formula	Penalty Term	Best For	Model Selection Rule
AIC	$-2l + 2q$	$2q$	General model comparison (large n)	Lower is better
AICc	$AIC + \frac{2q(q+1)}{n-q-1}$	Adjusted for small samples	Small sample sizes, simple random effects	Lower is better
BIC	$-2l + q\log(n)$	$q\log(n)$	Large samples, model selection in hypothesis testing	Lower is better

Key Takeaways

1. AIC is suitable for large datasets and general model comparisons but may favor overly complex models in small samples.
2. AICc corrects AIC's bias in small sample sizes.
3. BIC favors simpler models, especially as the sample size increases, making it suitable for hypothesis testing and situations where parsimony is essential.
4. Use MLE for comparing models with different fixed effects, and REML when comparing models with different random effects (same fixed effects).
5. When comparing random effects structures, AIC and BIC may not be reliable due to difficulty in counting effective parameters accurately.

8.4.4 Practical Example with Linear Mixed Models

Consider the Linear Mixed Model:

$$Y_{ik} = \begin{cases} \beta_0 + b_{1i} + (\beta_1 + b_{2i})t_{ij} + \epsilon_{ij} & L \\ \beta_0 + b_{1i} + (\beta_2 + b_{2i})t_{ij} + \epsilon_{ij} & H \\ \beta_0 + b_{1i} + (\beta_3 + b_{2i})t_{ij} + \epsilon_{ij} & C \end{cases}$$

where:

- $i = 1, \dots, N$ (subjects)
- $j = 1, \dots, n_i$ (repeated measures at time t_{ij})

$$\mathbf{Y}_i|b_i \sim N(\mathbf{X}_i\beta + \mathbf{1}b_i, \sigma^2\mathbf{I})$$
$$b_i \sim N(0, d_{11})$$

We aim to **estimate**:

- **Fixed effects**: β
- **Variance components**: σ^2, d_{11}
- **Random effects**: Predict b_i

When comparing models (e.g., different random slopes or covariance structures), we can compute:

- **AIC**: Penalizes model complexity with $2q$.
- **BIC**: Stronger penalty via $q\log(n)$, favoring simpler models.
- **AICc**: Adjusted AIC for small sample sizes.

```r
# Practical Example with Linear Mixed Models in R


# Load required libraries
library(lme4)     # For fitting linear mixed-effects models
library(MuMIn)    # For calculating AICc
library(dplyr)    # For data manipulation

# Set seed for reproducibility
set.seed(123)

# Simulate Data
N <- 50             # Number of subjects
n_i <- 5            # Number of repeated measures per subject
t_ij <- rep(1:n_i, N)  # Time points

# Treatment groups (L, H, C)
treatment <- rep(c("L", "H", "C"), length.out = N)
group <- factor(rep(treatment, each = n_i))

# Simulate random effects
b1_i <- rnorm(N, mean = 0, sd = 2)    # Random intercepts
b2_i <- rnorm(N, mean = 0, sd = 1)    # Random slopes

# Fixed effects
beta_0 <- 5
beta_1 <- 0.5
beta_2 <- 1
beta_3 <- 1.5

# Generate response variable Y based on the specified model
```

```r
Y <- numeric(N * n_i)
subject_id <- rep(1:N, each = n_i)

for (i in 1:N) {
    for (j in 1:n_i) {
        idx <- (i - 1) * n_i + j
        time <- t_ij[idx]

        # Treatment-specific model
        if (group[idx] == "L") {
            Y[idx] <-
                beta_0 + b1_i[i] + (beta_1 + b2_i[i]) * time + rnorm(1, 0, 1)
        } else if (group[idx] == "H") {
            Y[idx] <-
                beta_0 + b1_i[i] + (beta_2 + b2_i[i]) * time + rnorm(1, 0, 1)
        } else {
            Y[idx] <-
                beta_0 + b1_i[i] + (beta_3 + b2_i[i]) * time + rnorm(1, 0, 1)
        }
    }
}

# Combine into a data frame
data <- data.frame(
    Y = Y,
    time = t_ij,
    group = group,
    subject = factor(subject_id)
)

# Fit Linear Mixed Models
# Model 1: Random Intercepts Only
model1 <-
    lmer(Y ~ time * group + (1 | subject), data = data, REML = FALSE)

# Model 2: Random Intercepts and Random Slopes
model2 <-
    lmer(Y ~ time * group + (1 + time |
                                subject),
         data = data,
         REML = FALSE)

# Model 3: Simpler Model (No Interaction)
model3 <-
    lmer(Y ~ time + group + (1 | subject), data = data, REML = FALSE)
```

```r
# Extract Information Criteria
results <- data.frame(
    Model = c(
        "Random Intercepts",
        "Random Intercepts + Slopes",
        "No Interaction"
    ),
    AIC = c(AIC(model1), AIC(model2), AIC(model3)),
    BIC = c(BIC(model1), BIC(model2), BIC(model3)),
    AICc = c(AICc(model1), AICc(model2), AICc(model3))
)

# Display the results
print(results)
#>                         Model       AIC      BIC      AICc
#> 1          Random Intercepts 1129.2064 1157.378 1129.8039
#> 2 Random Intercepts + Slopes  974.5514 1009.766  975.4719
#> 3             No Interaction 1164.2797 1185.408 1164.6253
```

Interpretation of Results:

- Model 2 (with random intercepts and slopes) has the lowest AIC, BIC, and AICc, indicating the best fit among the models.
- Model 1 (random intercepts only) performs worse, suggesting that allowing random slopes improves model fit.
- Model 3 (simpler fixed effects without interaction) has the highest AIC/BIC/AICc, indicating poor fit compared to Models 1 and 2.

Model Selection Criteria:

Criterion	Best Model	Reason
AIC	Model 2	Best trade-off between fit and complexity
BIC	Model 2	Stronger penalty for complexity, still favored
AICc	Model 2	Adjusted for small samples, Model 2 still best

8.5 Split-Plot Designs

Split-plot designs are commonly used in experimental settings where there are **two or more factors**, and at least one of them requires **larger experimental units** compared to the other(s). This situation often arises in agricultural, industrial, and business experiments where certain treatments are harder or more expensive to apply.

Key Characteristics

- Two factors with different experimental unit requirements:
  - Factor A (Whole-plot factor): Requires large experimental units (e.g., different fields, production batches).
  - Factor B (Sub-plot factor): Can be applied to smaller units within the larger experimental units (e.g., plots within fields, products within batches).
- Blocking: The experiment is typically divided into blocks (or replicates) to account for variability. However, unlike Randomized Block Designs, the randomization process in split-plot designs occurs at two levels:
  - Whole-plot randomization: Factor A is randomized across large units within each block.
  - Sub-plot randomization: Factor B is randomized within each whole plot.

8.5.1 Example Setup

- **Factor A:** 3 levels (applied to large units).
- **Factor B:** 2 levels (applied within the large units).
- **4 Blocks (replicates):** Each containing all combinations of A and B.

Unlike in Randomized Block Designs, the randomization of Factor A is **restricted** due to the larger unit size. Factor A is applied **once per block**, while Factor B can be applied **multiple times within each block**.

8.5.2 Statistical Model for Split-Plot Designs

1. When Factor A is the Primary Focus (Whole-Plot Analysis)

$$Y_{ij} = \mu + \rho_i + \alpha_j + e_{ij}$$

Where:

- Y_{ij} = Response for the j-th level of factor A in the i-th block.
- μ = Overall mean.
- ρ_i = Random effect of the i-th block ($\rho_i \sim N(0, \sigma_\rho^2)$).
- α_j = Fixed effect of factor A (main effect).
- $e_{ij} \sim N(0, \sigma_e^2)$ = Whole-plot error (random), representing the variability within blocks due to factor A.

2. When Factor B is the Primary Focus (Sub-Plot Analysis)

$$Y_{ijk} = \mu + \phi_{ij} + \beta_k + \epsilon_{ijk}$$

Where:

- Y_{ijk} = Response for the k-th level of factor B within the j-th level of factor A and i-th block.
- ϕ_{ij} = Combined effect of block and factor A:

$$\phi_{ij} = \rho_i + \alpha_j + e_{ij}$$

- β_k = Fixed effect of factor B (main effect).
- $\epsilon_{ijk} \sim N(0, \sigma_\epsilon^2)$ = Sub-plot error (random), capturing variability within whole plots.

3. Full Split-Plot Model (Including Interaction)

$$Y_{ijk} = \mu + \rho_i + \alpha_j + e_{ij} + \beta_k + (\alpha\beta)_{jk} + \epsilon_{ijk}$$

Where:

- i = Block (replication). - j = Level of factor A (whole-plot factor).
- k = Level of factor B (sub-plot factor).
- μ = Overall mean.
- ρ_i = Random effect of the i-th block.
- α_j = Fixed main effect of factor A.
- $e_{ij} \sim N(0, \sigma_e^2)$ = Whole-plot error (random).
- β_k = Fixed main effect of factor B.
- $(\alpha\beta)_{jk}$ = Fixed interaction effect between factors A and B.
- $\epsilon_{ijk} \sim N(0, \sigma_\epsilon^2)$ = Sub-plot error (random).

8.5.3 Approaches to Analyzing Split-Plot Designs

1. ANOVA Perspective

Whole-Plot Comparisons:

- Factor A vs. Whole-Plot Error:
  Compare the variation due to factor A (α_j) against the whole-plot error (e_{ij}).

- Blocks vs. Whole-Plot Error:
  Compare the variation due to blocks (ρ_i) against the whole-plot error (e_{ij}).

Sub-Plot Comparisons:

- Factor B vs. Sub-Plot Error:
  Compare the variation due to factor B (β_k) against the sub-plot error (ϵ_{ijk}).
- Interaction (A $\times$ B) vs. Sub-Plot Error:
  Compare the interaction effect ($(\alpha\beta)_{jk}$) against the sub-plot error (ϵ_{ijk}).

2. **Mixed Model Perspective**

A more flexible approach is to treat split-plot designs using mixed-effects models, which can handle both fixed and random effects explicitly:

$$\mathbf{Y} = \mathbf{X} \ + \mathbf{Zb} +$$

Where:

- $\mathbf{Y}$ = Vector of observed responses.
- $\mathbf{X}$ = Design matrix for fixed effects (e.g., factors A, B, and their interaction).
- β = Vector of fixed-effect coefficients (e.g., μ, α_j, β_k, $(\alpha\beta)_{jk}$).
- $\mathbf{Z}$ = Design matrix for random effects (e.g., blocks and whole-plot errors).
- $\mathbf{b}$ = Vector of random-effect coefficients (e.g., ρ_i, e_{ij}).
- ϵ = Vector of residuals (sub-plot error).

Mixed models are particularly useful when:

- There are **unbalanced designs** (missing data).
- You need to account for **complex correlation structures** within the data.

8.5.4 Application: Split-Plot Design

Consider an agricultural experiment designed to study the effects of irrigation and crop variety on yield. This scenario is well-suited for a split-plot design because irrigation treatments are applied to large plots (fields), while different crop varieties are planted within these plots.

8.5.4.1 Model Specification

The linear mixed-effects model is defined as:

$$y_{ijk} = \mu + i_i + v_j + (iv)_{ij} + f_k + \epsilon_{ijk}$$

Where:

- y_{ijk} = Observed yield for the i-th irrigation, j-th variety, in the k-th field.
- μ = Overall mean yield.
- i_i = Fixed effect of the i-th irrigation level.
- v_j = Fixed effect of the j-th crop variety.
- $(iv)_{ij}$ = Interaction effect between irrigation and variety (fixed).
- $f_k \sim N(0, \sigma_f^2)$ = Random effect of field (captures variability between fields).
- $\epsilon_{ijk} \sim N(0, \sigma_\epsilon^2)$ = Residual error.

Note:
Since each variety-field combination is observed only once, we cannot model a random interaction between variety and field.

8.5.4.2 Data Exploration

```
library(ggplot2)
data(irrigation, package = "faraway")  # Load the dataset

# Summary statistics and preview
summary(irrigation)
#>      field   irrigation variety     yield
#>  f1     :2   i1:4       v1:8    Min.   :34.80
#>  f2     :2   i2:4       v2:8    1st Qu.:37.60
#>  f3     :2   i3:4               Median :40.15
#>  f4     :2   i4:4               Mean   :40.23
#>  f5     :2                      3rd Qu.:42.73
#>  f6     :2                      Max.   :47.60
#>  (Other):4
head(irrigation, 4)
#>   field irrigation variety yield
#> 1    f1         i1      v1  35.4
#> 2    f1         i1      v2  37.9
#> 3    f2         i2      v1  36.7
#> 4    f2         i2      v2  38.2

# Exploratory plot: Yield by field, colored by variety and shaped by irrigation
ggplot(irrigation,
       aes(
           x     = field,
           y     = yield,
```

```
            shape = irrigation,
            color = variety
        )) +
    geom_point(size = 3) +
    labs(title = "Yield by Field, Irrigation, and Variety",
         x = "Field",
         y = "Yield") +
    theme_minimal()
```

This plot helps visualize how yield varies across fields, under different irrigation treatments, and for different varieties.

8.5.4.3 Fitting the Initial Mixed-Effects Model

We fit a mixed-effects model where:

- Irrigation and variety (and their interaction) are fixed effects.
- Field is modeled as a random effect to account for variability between fields.

```
library(lmerTest)  # Provides p-values for lmer models

# Full model with interaction term
sp_model <-
    lmer(yield ~ irrigation * variety + (1 | field), data = irrigation)
summary(sp_model)
#> Linear mixed model fit by REML. t-tests use Satterthwaite's method [
#> lmerModLmerTest]
```

```
#> Formula: yield ~ irrigation * variety + (1 | field)
#>    Data: irrigation
#>
#> REML criterion at convergence: 45.4
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -0.7448 -0.5509  0.0000  0.5509  0.7448
#>
#> Random effects:
#>  Groups   Name        Variance Std.Dev.
#>  field    (Intercept) 16.200   4.025
#>  Residual              2.107   1.452
#> Number of obs: 16, groups:  field, 8
#>
#> Fixed effects:
#>                       Estimate Std. Error     df t value Pr(>|t|)
#> (Intercept)             38.500      3.026  4.487  12.725 0.000109 ***
#> irrigationi2             1.200      4.279  4.487   0.280 0.791591
#> irrigationi3             0.700      4.279  4.487   0.164 0.877156
#> irrigationi4             3.500      4.279  4.487   0.818 0.454584
#> varietyv2                0.600      1.452  4.000   0.413 0.700582
#> irrigationi2:varietyv2  -0.400      2.053  4.000  -0.195 0.855020
#> irrigationi3:varietyv2  -0.200      2.053  4.000  -0.097 0.927082
#> irrigationi4:varietyv2   1.200      2.053  4.000   0.584 0.590265
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Correlation of Fixed Effects:
#>             (Intr) irrgt2 irrgt3 irrgt4 vrtyv2 irr2:2 irr3:2
#> irrigation2 -0.707
#> irrigation3 -0.707  0.500
#> irrigation4 -0.707  0.500  0.500
#> varietyv2   -0.240  0.170  0.170  0.170
#> irrgtn2:vr2  0.170 -0.240 -0.120 -0.120 -0.707
#> irrgtn3:vr2  0.170 -0.120 -0.240 -0.120 -0.707  0.500
#> irrgtn4:vr2  0.170 -0.120 -0.120 -0.240 -0.707  0.500  0.500

# ANOVA table using Kenward-Roger approximation for accurate p-values
anova(sp_model, ddf = "Kenward-Roger")
#> Type III Analysis of Variance Table with Kenward-Roger's method
#>                    Sum Sq Mean Sq NumDF DenDF F value Pr(>F)
#> irrigation         2.4545 0.81818     3     4  0.3882 0.7685
#> variety            2.2500 2.25000     1     4  1.0676 0.3599
#> irrigation:variety 1.5500 0.51667     3     4  0.2452 0.8612
```

- Check the p-value of the interaction term (`irrigation:variety`).
  - If insignificant, this suggests no strong evidence of an interaction effect, and we may simplify the model by removing it.

8.5.4.4 Model Simplification: Testing for Additivity

We compare the **full model** (with interaction) to an **additive model** (without interaction):

```
library(lme4)

# Additive model (no interaction)
sp_model_additive <- lmer(yield ~ irrigation + variety + (1 | field), data = irrigatio

# Likelihood ratio test comparing the two models
anova(sp_model_additive, sp_model, ddf = "Kenward-Roger")
#> Data: irrigation
#> Models:
#> sp_model_additive: yield ~ irrigation + variety + (1 | field)
#> sp_model: yield ~ irrigation * variety + (1 | field)
#>                   npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
#> sp_model_additive    7 83.959 89.368 -34.980   69.959
#> sp_model            10 88.609 96.335 -34.305   68.609 1.3503  3     0.7172
```

- **Hypotheses:**
  - H_0: The **additive model** (without interaction) fits the data adequately.
  - H_a: The **interaction model** provides a significantly better fit.
- **Interpretation:**
  - If the p-value is insignificant, we fail to reject H_0, meaning the simpler additive model is sufficient.
  - Check AIC and BIC: Lower values indicate a better-fitting model, supporting the use of the additive model if consistent with the hypothesis test.

8.5.4.5 Assessing the Random Effect: Exact Restricted Likelihood Ratio Test

To verify whether the random field effect is necessary, we conduct an exact RLRT:

- Hypotheses:
  - H_0: $\sigma_f^2 = 0$ (no variability between fields; random effect is unnecessary).

- H_a: $\sigma_f^2 > 0$ (random field effect is significant).

```
library(RLRsim)

# RLRT for the random effect of field
exactRLRT(sp_model)
#>
#>  simulated finite sample distribution of RLRT.
#>
#>  (p-value based on 10000 simulated values)
#>
#> data:
#> RLRT = 6.1118, p-value = 0.01
```

Interpretation:

- If the p-value is significant, we reject H_0, confirming that the random field effect is essential.
- A significant random effect implies substantial variability between fields that must be accounted for in the model.

8.6 Repeated Measures in Mixed Models

Repeated measures data arise when **multiple observations** are collected from the **same subjects** over time or under different conditions. This introduces **correlation** between observations from the same subject, which must be accounted for in the statistical model.

Mixed-effects models are particularly effective for repeated measures because they allow us to model both **fixed effects** (e.g., treatment, time) and **random effects** (e.g., subject-specific variability).

The general form of a mixed-effects model for repeated measures is:

$$Y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \delta_{i(k)} + \epsilon_{ijk}$$

Where:

- Y_{ijk} = Response for the i-th group, j-th time point, and k-th subject.
- μ = Overall mean.
- α_i = Fixed effect of the i-th group (e.g., treatment group).
- β_j = Fixed effect of the j-th time point (repeated measure effect).
- $(\alpha\beta)_{ij}$ = Interaction effect between group and time (fixed).

- $\delta_{i(k)} \sim N(0, \sigma_\delta^2)$ = Random effect of the k-th subject within the i-th group (captures subject-specific deviations).
- $\epsilon_{ijk} \sim N(0, \sigma^2)$ = Residual error (independent across observations).

Here, $i = 1, \dots, n_A$ (number of groups), $j = 1, \dots, n_B$ (number of repeated measures), and $k = 1, \dots, n_i$ (number of subjects in group i).

The **variance-covariance matrix** of the repeated observations for the k-th subject in the i-th group is given by:

$$\mathbf{Y}_{ik} = \begin{pmatrix} Y_{i1k} \\ Y_{i2k} \\ \vdots \\ Y_{in_Bk} \end{pmatrix}$$

1. Compound Symmetry (CS) Structure

Under the **compound symmetry** assumption (common in random-intercepts models), the covariance matrix is:

$$_{\text{subject}} = \begin{pmatrix} \sigma_\delta^2 + \sigma^2 & \sigma_\delta^2 & \cdots & \sigma_\delta^2 \\ \sigma_\delta^2 & \sigma_\delta^2 + \sigma^2 & \cdots & \sigma_\delta^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_\delta^2 & \sigma_\delta^2 & \cdots & \sigma_\delta^2 + \sigma^2 \end{pmatrix}$$

This matrix can be rewritten as:

$$_{\text{subject}} = (\sigma_\delta^2 + \sigma^2) \begin{pmatrix} 1 & \rho & \cdots & \rho \\ \rho & 1 & \cdots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \cdots & 1 \end{pmatrix}$$

Where:

- σ_δ^2 = Variance due to subject-specific random effects.
- σ^2 = Residual variance.
- $\rho = \frac{\sigma_\delta^2}{\sigma_\delta^2 + \sigma^2}$ = Intra-class correlation coefficient (ICC).

Key Points:

- Compound Symmetry Structure is the product of a scalar and a correlation matrix.
- The correlation between any two repeated measures from the same subject is constant (ρ).

- This structure assumes equal correlation across time points, which may not hold if measurements are collected over time.

Refer to Random-Intercepts Model for a detailed discussion of compound symmetry.

---

2. Autoregressive (AR(1)) Structure

If repeated measures are collected over time, it may be more appropriate to assume an autoregressive correlation structure, where correlations decay as the time gap increases.

The AR(1) variance-covariance matrix is:

$$_{\text{subject}} = \sigma^2 \begin{pmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{n_B-1} \\ \rho & 1 & \rho & \cdots & \rho^{n_B-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{n_B-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{n_B-1} & \rho^{n_B-2} & \rho^{n_B-3} & \cdots & 1 \end{pmatrix}$$

Where:

- σ^2 = Residual variance.
- ρ = Autoregressive parameter ($|\rho| < 1$), representing the correlation between consecutive time points.

Key Characteristics:

- Correlations decrease exponentially as the time lag increases.
- Appropriate for longitudinal data where temporal proximity influences correlation.

In matrix notation, the mixed model can be written as:

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon$$

Where:

- $\mathbf{Y}$ = Vector of observed responses.
- $\mathbf{X}$ = Design matrix for fixed effects (e.g., group, time, interaction).
- β = Vector of fixed-effect coefficients.
- $\epsilon \sim N(0, \sigma^2\ )$ = Random error vector.
- = Variance-covariance matrix of residuals:

  - Block diagonal structure if the covariance structure is identical for each subject.
  - Within each block (subject), the structure can be compound symmetry, AR(1), or another suitable structure depending on the data.

Choosing the Right Covariance Structure

- Compound Symmetry:
  - Suitable when correlations are constant across repeated measures (e.g., in randomized controlled trials).
  - Simple and interpretable but may be too restrictive for longitudinal data.
- Autoregressive (AR(1)):
  - Best when measurements are taken over equally spaced time intervals and correlations decay over time.
  - Assumes stronger correlation for adjacent time points.
- Unstructured (UN):
  - Allows different variances and covariances for each time point.
  - Provides maximum flexibility but requires more parameters and larger sample sizes.

Model selection criteria (AIC, BIC, likelihood ratio tests) can help determine the most appropriate covariance structure.

8.7 Unbalanced or Unequally Spaced Data

In many real-world applications, data are **unbalanced** (i.e., different numbers of observations per subject) or **unequally spaced** over time. This is common in longitudinal studies, clinical trials, and business analytics where subjects may be observed at irregular intervals or miss certain time points.

Mixed-effects models are flexible enough to handle such data structures, especially when we carefully model the **variance-covariance structure** of repeated measurements.

Consider the following mixed-effects model:

$$Y_{ikt} = \beta_0 + \beta_{0i} + \beta_1 t + \beta_{1i} t + \beta_2 t^2 + \beta_{2i} t^2 + \epsilon_{ikt}$$

Where:

- Y_{ikt} = Response for the k-th subject in the i-th group at time t.
- $i = 1, 2$ = Groups (e.g., treatment vs. control).
- $k = 1, \dots, n_i$ = Individuals within group i.

- $t = (t_1, t_2, t_3, t_4)$ = Time points (which may be unequally spaced).

Model Components:

- **Fixed Effects:**
  - β_0 = Overall intercept (baseline).
  - β_1 = Common linear time trend.
  - β_2 = Common quadratic time trend.
- **Random Effects:**
  - β_{0i} = Random intercept for group i (captures group-specific baseline variation).
  - β_{1i} = Random slope for time in group i (captures group-specific linear trends).
  - β_{2i} = Random quadratic effect for group i (captures group-specific curvature over time).
- **Residual Error:**
  - $\epsilon_{ikt} \sim N(0, \sigma^2)$ = Measurement error, assumed independent of the random effects.

8.7.1 Variance-Covariance Structure: Power Model

Since observations are taken at unequally spaced time points, we cannot rely on simple structures like compound symmetry or AR(1). Instead, we use a power covariance model, which allows the correlation to depend on the distance between time points.

The variance-covariance matrix of the repeated measurements for subject k in group i is:

$$_{ik} = \sigma^2 \begin{pmatrix} 1 & \rho^{|t_2-t_1|} & \rho^{|t_3-t_1|} & \rho^{|t_4-t_1|} \\ \rho^{|t_2-t_1|} & 1 & \rho^{|t_3-t_2|} & \rho^{|t_4-t_2|} \\ \rho^{|t_3-t_1|} & \rho^{|t_3-t_2|} & 1 & \rho^{|t_4-t_3|} \\ \rho^{|t_4-t_1|} & \rho^{|t_4-t_2|} & \rho^{|t_4-t_3|} & 1 \end{pmatrix}$$

Where:

- σ^2 = Residual variance.
- ρ = Correlation parameter ($0 < |\rho| < 1$), controlling how correlation decays with increasing time gaps.
- $|t_j - t_i|$ = Absolute time difference between measurements at times t_i and t_j.

Key Characteristics:

- The correlation between observations decreases as the time difference increases, similar to AR(1), but flexible enough to handle unequal time intervals.
- This structure is sometimes referred to as a continuous-time autoregressive model or power covariance model.

After fitting the full model, we can evaluate whether all terms are necessary, focusing on the random effects:

- β_{0i} (Random Intercepts):
  Is there significant baseline variability between groups?
- β_{1i} (Random Slopes for Time):
  Do groups exhibit different linear trends over time?
- β_{2i} (Random Quadratic Terms):
  Is there group-specific curvature in the response over time?

Model Comparison Approach:

1. **Fit the Full Model:**
   Includes all random effects.
2. **Fit Reduced Models:**
   Systematically remove random effects (e.g., quadratic terms) to create simpler models.
3. **Compare Models Using:**
   - **Likelihood Ratio Tests (LRT):**
     Test whether the more complex model significantly improves fit.
   - **Information Criteria (AIC, BIC):**
     Lower values indicate a better trade-off between fit and complexity.
4. **Assess Random Effects:**
   - Use the **exactRLRT** test to determine if random effects are significant.
   - Check variance estimates: if the variance of a random effect is near zero, it may not be necessary.

In matrix notation, the model can be written as:

$$\mathbf{Y} = \mathbf{X}\beta + \mathbf{Zb} + \epsilon$$

Where:

- $\mathbf{Y}$ = Vector of observed responses.

- $\mathbf{X}$ = Design matrix for fixed effects (intercept, time, time^2, and group interactions).
- β = Vector of fixed-effect coefficients.
- $\mathbf{Z}$ = Design matrix for random effects (random intercepts, slopes, etc.).
- $\mathbf{b} \sim N(0, \mathbf{G})$ = Vector of random effects with covariance matrix $\mathbf{G}$.
- $\epsilon \sim N(0, \mathbf{R})$ = Vector of residual errors with covariance matrix $\mathbf{R}$, where $\mathbf{R}$ follows the power covariance structure.

8.8 Application: Mixed Models in Practice

Several R packages are available for fitting mixed-effects models, each with unique strengths:

- `nlme`
  - Supports nested and crossed random effects.
  - Flexible for complex covariance structures.
  - Less intuitive syntax compared to `lme4`.
- `lme4`
  - Computationally efficient and widely used.
  - User-friendly formula syntax.
  - Can handle non-normal responses (e.g., GLMMs).
  - For detailed documentation, refer to Bates et al. (2014).
- Others:
  - Bayesian Mixed Models: `MCMCglmm`, `brms`.
  - Genetics/Plant Breeding: `ASReml`.

8.8.1 Example 1: Pulp Brightness Analysis

8.8.1.1 Model Specification

We start with a random-intercepts model for pulp brightness:

$$y_{ij} = \mu + \alpha_i + \epsilon_{ij}$$

Where:

- $i = 1, \dots, a$ = Groups for random effect α_i.
- $j = 1, \dots, n$ = Observations per group.
- μ = Overall mean brightness (fixed effect).
- $\alpha_i \sim N(0, \sigma_\alpha^2)$ = Group-specific random effect.

- $\epsilon_{ij} \sim N(0, \sigma_\epsilon^2)$ = Residual error.

This implies a compound symmetry structure, where the intraclass correlation coefficient is:

$$\rho = \frac{\sigma_\alpha^2}{\sigma_\alpha^2 + \sigma_\epsilon^2}$$

- If $\sigma_\alpha^2 \to 0$: Low correlation within groups ($\rho \to 0$) (i.e., when factor a doesn't explain much variation).
- If $\sigma_\alpha^2 \to \infty$: High correlation within groups ($\rho \to 1$).

8.8.1.2 Data Exploration

```r
data(pulp, package = "faraway")
library(ggplot2)
library(dplyr)

# Visualize brightness by operator
ggplot(pulp, aes(x = operator, y = bright)) +
    geom_boxplot(fill = "lightblue") +
    labs(title = "Pulp Brightness by Operator",
         x = "Operator",
         y = "Brightness") +
    theme_minimal()
```

```r
# Group-wise summary
```

```r
pulp %>%
    group_by(operator) %>%
    summarise(average_brightness = mean(bright), .groups = 'drop')
#> # A tibble: 4 x 2
#>   operator average_brightness
#>   <fct>                 <dbl>
#> 1 a                      60.2
#> 2 b                      60.1
#> 3 c                      60.6
#> 4 d                      60.7
```

8.8.1.3 Fitting the Mixed Model with `lme4`

```r
library(lme4)

# Random intercepts model: operator as a random effect
mixed_model <- lmer(bright ~ 1 + (1 | operator), data = pulp)

# Model summary
summary(mixed_model)
#> Linear mixed model fit by REML. t-tests use Satterthwaite's method [
#> lmerModLmerTest]
#> Formula: bright ~ 1 + (1 | operator)
#>    Data: pulp
#>
#> REML criterion at convergence: 18.6
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -1.4666 -0.7595 -0.1244  0.6281  1.6012
#>
#> Random effects:
#>  Groups   Name        Variance Std.Dev.
#>  operator (Intercept) 0.06808  0.2609
#>  Residual             0.10625  0.3260
#> Number of obs: 20, groups:  operator, 4
#>
#> Fixed effects:
#>             Estimate Std. Error      df t value Pr(>|t|)
#> (Intercept)  60.4000     0.1494  3.0000   404.2 3.34e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Fixed effects (overall mean)
fixef(mixed_model)
```

```
#> (Intercept)
#>        60.4

# Random effects (BLUPs)
ranef(mixed_model)
#> $operator
#>   (Intercept)
#> a  -0.1219403
#> b  -0.2591231
#> c   0.1676679
#> d   0.2133955
#>
#> with conditional variances for "operator"

# Variance components
VarCorr(mixed_model)
#>  Groups   Name        Std.Dev.
#>  operator (Intercept) 0.26093
#>  Residual             0.32596
re_dat <- as.data.frame(VarCorr(mixed_model))

# Intraclass Correlation Coefficient
rho <- re_dat[1, 'vcov'] / (re_dat[1, 'vcov'] + re_dat[2, 'vcov'])
rho
#> [1] 0.3905354
```

8.8.1.4 Inference with `lmerTest`

To obtain p-values for fixed effects using Satterthwaite's approximation:

```
library(lmerTest)

# Model with p-values
summary(lmer(bright ~ 1 + (1 | operator), data = pulp))$coefficients
#>             Estimate Std. Error df  t value     Pr(>|t|)
#> (Intercept)     60.4  0.1494434  3 404.1664 3.340265e-08

# Confidence interval for the fixed effect
confint(mixed_model)[3,]
#>   2.5 %  97.5 %
#> 60.0713 60.7287
```

In this example, we can see that the confidence interval computed by `confint` from the `lme4` package is very close to the one computed by `confint` from the `lmerTest` package.

8.8.1.5 Bayesian Mixed Model with `MCMCglmm`

```
library(MCMCglmm)

# Bayesian mixed model
mixed_model_bayes <- MCMCglmm(
  bright ~ 1,
  random = ~ operator,
  data = pulp,
  verbose = FALSE
)

# Posterior summaries
summary(mixed_model_bayes)$solutions
#>             post.mean l-95% CI u-95% CI eff.samp pMCMC
#> (Intercept)  60.40789  60.1488 60.69595     1000 0.001
```

Bayesian credible intervals may differ slightly from frequentist confidence intervals due to prior assumptions.

8.8.1.6 Predictions

```
# Random effects predictions (BLUPs)
ranef(mixed_model)$operator
#>   (Intercept)
#> a  -0.1219403
#> b  -0.2591231
#> c   0.1676679
#> d   0.2133955

# Predictions per operator
fixef(mixed_model) + ranef(mixed_model)$operator
#>   (Intercept)
#> a    60.27806
#> b    60.14088
#> c    60.56767
#> d    60.61340

# Equivalent using predict()
predict(mixed_model, newdata = data.frame(operator = c('a', 'b', 'c', 'd')))
#>        1        2        3        4
#> 60.27806 60.14088 60.56767 60.61340
```

For **bootstrap confidence intervals**, use:

```
bootMer(mixed_model, FUN = fixef, nsim = 100)
#>
#> PARAMETRIC BOOTSTRAP
#>
#>
#> Call:
#> bootMer(x = mixed_model, FUN = fixef, nsim = 100)
#>
#>
#> Bootstrap Statistics :
#>     original        bias    std. error
#> t1*     60.4 -0.0005452538    0.156374
```

8.8.2 Example 2: Penicillin Yield (GLMM with Blocking)

```
data(penicillin, package = "faraway")
library(ggplot2)

# Visualize yield by treatment and blend
ggplot(penicillin, aes(y = yield, x = treat, shape = blend, color = blend)) +
  geom_point(size = 3) +
  labs(title = "Penicillin Yield by Treatment and Blend") +
  theme_minimal()
```

```
# Mixed model: blend as random effect, treatment as fixed
mixed_model <- lmer(yield ~ treat + (1 | blend), data = penicillin)
```

```r
summary(mixed_model)
#> Linear mixed model fit by REML. t-tests use Satterthwaite's method [
#> lmerModLmerTest]
#> Formula: yield ~ treat + (1 | blend)
#>    Data: penicillin
#>
#> REML criterion at convergence: 103.8
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -1.4152 -0.5017 -0.1644  0.6830  1.2836
#>
#> Random effects:
#>  Groups   Name        Variance Std.Dev.
#>  blend    (Intercept) 11.79    3.434
#>  Residual             18.83    4.340
#> Number of obs: 20, groups:  blend, 5
#>
#> Fixed effects:
#>             Estimate Std. Error     df t value Pr(>|t|)
#> (Intercept)   84.000      2.475 11.075  33.941 1.51e-12 ***
#> treatB         1.000      2.745 12.000   0.364   0.7219
#> treatC         5.000      2.745 12.000   1.822   0.0935 .
#> treatD         2.000      2.745 12.000   0.729   0.4802
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Correlation of Fixed Effects:
#>        (Intr) treatB treatC
#> treatB -0.555
#> treatC -0.555  0.500
#> treatD -0.555  0.500  0.500

# BLUPs for each blend
ranef(mixed_model)$blend
#>        (Intercept)
#> Blend1   4.2878788
#> Blend2  -2.1439394
#> Blend3  -0.7146465
#> Blend4   1.4292929
#> Blend5  -2.8585859
```

Testing for Treatment Effect

```r
# ANOVA for fixed effects
anova(mixed_model)
```

```
#> Type III Analysis of Variance Table with Satterthwaite's method
#>       Sum Sq Mean Sq NumDF DenDF F value Pr(>F)
#> treat     70  23.333     3    12  1.2389 0.3387
```

Since the p-value $> .05$, we fail to reject the null hypothesis (no treatment effect).

Model Comparison with Kenward-Roger Approximation

```
library(pbkrtest)

# Full model vs. null model
full_model <-
    lmer(yield ~ treat + (1 | blend), penicillin, REML = FALSE)
null_model <-
    lmer(yield ~ 1 + (1 | blend), penicillin, REML = FALSE)

# Kenward-Roger approximation
KRmodcomp(full_model, null_model)
#> large : yield ~ treat + (1 | blend)
#> small : yield ~ 1 + (1 | blend)
#>          stat     ndf     ddf F.scaling p.value
#> Ftest  1.2389  3.0000 12.0000         1  0.3387
```

The results are consistent with the earlier ANOVA: **no significant treatment effect**.

8.8.3 Example 3: Growth in Rats Over Time

```
rats <- read.csv(
    "images/rats.dat",
    header = FALSE,
    sep = ' ',
    col.names = c('Treatment', 'rat', 'age', 'y')
)

# Log-transformed time variable
rats$t <- log(1 + (rats$age - 45) / 10)
```

Model Fitting

```
# Treatment as fixed effect, random intercepts for rats
rat_model <- lmer(y ~ t:Treatment + (1 | rat), data = rats)
summary(rat_model)
#> Linear mixed model fit by REML. t-tests use Satterthwaite's method [
#> lmerModLmerTest]
#> Formula: y ~ t:Treatment + (1 | rat)
```

```
#>    Data: rats
#>
#> REML criterion at convergence: 932.4
#>
#> Scaled residuals:
#>      Min       1Q   Median       3Q      Max
#> -2.25574 -0.65898 -0.01163  0.58356  2.88309
#>
#> Random effects:
#>  Groups   Name        Variance Std.Dev.
#>  rat      (Intercept) 3.565    1.888
#>  Residual             1.445    1.202
#> Number of obs: 252, groups:  rat, 50
#>
#> Fixed effects:
#>                Estimate Std. Error       df t value Pr(>|t|)
#> (Intercept)     68.6074     0.3312  89.0275  207.13   <2e-16 ***
#> t:Treatmentcon   7.3138     0.2808 247.2762   26.05   <2e-16 ***
#> t:Treatmenthig   6.8711     0.2276 247.7097   30.19   <2e-16 ***
#> t:Treatmentlow   7.5069     0.2252 247.5196   33.34   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Correlation of Fixed Effects:
#>             (Intr) t:Trtmntc t:Trtmnth
#> t:Tretmntcn -0.327
#> t:Tretmnthg -0.340  0.111
#> t:Tretmntlw -0.351  0.115     0.119
anova(rat_model)
#> Type III Analysis of Variance Table with Satterthwaite's method
#>             Sum Sq Mean Sq NumDF  DenDF F value    Pr(>F)
#> t:Treatment 3181.9  1060.6     3 223.21  734.11 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Since the **p-value is significant**, we conclude that the **treatment effect varies over time**.

8.8.4 Example 4: Tree Water Use (Agridat)

```
library(agridat)
dat <- harris.wateruse

# Visualizing water use by species and age
library(latticeExtra)
```

```
useOuterStrips(
  xyplot(water ~ day | species * age,
         dat, group = tree,
         type = c('p', 'smooth'),
         main = "harris.wateruse 2 species, 2 ages (10 trees each)",
         as.table = TRUE)
)
```

Remove outlier

```
dat <- subset(dat, day!=268)
```

Plot between water use and day for one age and species group

```
xyplot(
    water ~ day | tree,
    dat,
    subset   = age == "A2" & species == "S2",
    as.table = TRUE,
    type     = c('p', 'smooth'),
    ylab     = "Water use profiles of individual trees",
    main     = "harris.wateruse (Age 2, Species 2)"
)
```

```
# Rescale day for nicer output, and convergence issues
dat <- transform(dat, ti = day / 100)
# add quadratic term
dat <- transform(dat, ti2 = ti * ti)

# Start with a subgroup: age 2, species 2
d22 <- droplevels(subset(dat, age == "A2" & species == "S2"))
```

Fitting with `nlme` using `lme`

```
library(nlme)

## We use pdDiag() to get uncorrelated random effects
m1n <- lme(
    water ~ 1 + ti + ti2,
    #intercept, time and time-squared = fixed effects
    data = d22,
    na.action = na.omit,
    random = list(tree = pdDiag(~ 1 + ti + ti2))
    # random intercept, time
    # and time squared per tree = random effects
)

# for all trees
# m1n <- lme(
#   water ~ 1 + ti + ti2,
#   random = list(tree = pdDiag(~ 1 + ti + ti2)),
#   data = dat,
```

```
#   na.action = na.omit
# )

summary(m1n)
#> Linear mixed-effects model fit by REML
#>   Data: d22
#>        AIC     BIC    logLik
#>   276.5142 300.761 -131.2571
#>
#> Random effects:
#>  Formula: ~1 + ti + ti2 | tree
#>  Structure: Diagonal
#>         (Intercept)           ti          ti2  Residual
#> StdDev:   0.5187869 1.631223e-05 4.374982e-06 0.3836614
#>
#> Fixed effects:  water ~ 1 + ti + ti2
#>                  Value Std.Error  DF   t-value p-value
#> (Intercept) -10.798799 0.8814666 227 -12.25094       0
#> ti           12.346704 0.7827112 227  15.77428       0
#> ti2          -2.838503 0.1720614 227 -16.49704       0
#>  Correlation:
#>     (Intr) ti
#> ti  -0.979
#> ti2  0.970 -0.997
#>
#> Standardized Within-Group Residuals:
#>         Min          Q1         Med          Q3         Max
#> -3.07588246 -0.58531056  0.01210209  0.65402695  3.88777402
#>
#> Number of Observations: 239
#> Number of Groups: 10

ranef(m1n)
#>     (Intercept)            ti           ti2
#> T04   0.1985796  2.070606e-09  6.397103e-10
#> T05   0.3492827  3.199664e-10 -6.211457e-11
#> T19  -0.1978989 -9.879555e-10 -2.514502e-10
#> T23   0.4519003 -4.206418e-10 -3.094113e-10
#> T38  -0.6457494 -2.069198e-09 -4.227912e-10
#> T40   0.3739432  4.199061e-10 -3.260161e-11
#> T49   0.8620648  1.160387e-09 -6.925457e-12
#> T53  -0.5655049 -1.064849e-09 -5.870462e-11
#> T67  -0.4394623 -4.482549e-10  2.752922e-11
#> T71  -0.3871552  1.020034e-09  4.767595e-10
```

```r
fixef(m1n)
#> (Intercept)          ti         ti2
#>  -10.798799   12.346704   -2.838503
```

Fitting with `lme4` using `lmer`

```r
library(lme4)

m1lmer <-
    lmer(water ~ 1 + ti + ti2 + (ti + ti2 ||
                                    tree),
         data = d22,
         na.action = na.omit)

# for all trees
# m1lmer <- lmer(water ~ 1 + ti + ti2 + (ti + ti2 || tree),
#                data = dat, na.action = na.omit)

# summary(m1lmer)
ranef(m1lmer)
#> $tree
#>     (Intercept) ti ti2
#> T04   0.1985796  0   0
#> T05   0.3492827  0   0
#> T19  -0.1978989  0   0
#> T23   0.4519003  0   0
#> T38  -0.6457494  0   0
#> T40   0.3739432  0   0
#> T49   0.8620648  0   0
#> T53  -0.5655049  0   0
#> T67  -0.4394623  0   0
#> T71  -0.3871552  0   0
#>
#> with conditional variances for "tree"

fixef(m1lmer)
#> (Intercept)          ti         ti2
#>  -10.798799   12.346704   -2.838503
```

Notes:

- `||` double pipes = uncorrelated random effects
- To remove the intercept term:
  - `(0+ti|tree)`
  - `(ti-1|tree)`

```
m1l <-
    lmer(water ~ 1 + ti + ti2
         + (1 | tree) + (0 + ti | tree)
         + (0 + ti2 | tree), data = d22)
ranef(m1l)
#> $tree
#>     (Intercept) ti ti2
#> T04   0.1985796  0   0
#> T05   0.3492827  0   0
#> T19  -0.1978989  0   0
#> T23   0.4519003  0   0
#> T38  -0.6457494  0   0
#> T40   0.3739432  0   0
#> T49   0.8620648  0   0
#> T53  -0.5655049  0   0
#> T67  -0.4394623  0   0
#> T71  -0.3871552  0   0
#>
#> with conditional variances for "tree"
fixef(m1l)
#> (Intercept)          ti         ti2
#>  -10.798799   12.346704   -2.838503
```

Adding Correlation Structure

```
m2n <- lme(
    water ~ 1 + ti + ti2,
    data = d22,
    random = ~ 1 | tree,
    cor = corExp(form =  ~ day | tree),
    na.action = na.omit
)

# for all trees
# m2n <- lme(
#   water ~ 1 + ti + ti2,
#   random = ~ 1 | tree,
#   cor = corExp(form = ~ day | tree),
#   data = dat,
#   na.action = na.omit
# )

summary(m2n)
#> Linear mixed-effects model fit by REML
#>   Data: d22
#>        AIC      BIC   logLik
```

```
#>   263.3081 284.0911 -125.654
#>
#> Random effects:
#>  Formula: ~1 | tree
#>         (Intercept)  Residual
#> StdDev:   0.5154042 0.3925777
#>
#> Correlation Structure: Exponential spatial correlation
#>  Formula: ~day | tree
#>  Parameter estimate(s):
#>    range
#> 3.794624
#> Fixed effects:  water ~ 1 + ti + ti2
#>                  Value Std.Error  DF   t-value p-value
#> (Intercept) -11.223310 1.0988725 227 -10.21348       0
#> ti           12.712094 0.9794235 227  12.97916       0
#> ti2          -2.913682 0.2148551 227 -13.56115       0
#>  Correlation:
#>     (Intr) ti
#> ti  -0.985
#> ti2  0.976 -0.997
#>
#> Standardized Within-Group Residuals:
#>         Min          Q1         Med          Q3         Max
#> -3.04861039 -0.55703950  0.00278101  0.62558762  3.80676991
#>
#> Number of Observations: 239
#> Number of Groups: 10
ranef(m2n)
#>     (Intercept)
#> T04   0.1929971
#> T05   0.3424631
#> T19  -0.1988495
#> T23   0.4538660
#> T38  -0.6413664
#> T40   0.3769378
#> T49   0.8410043
#> T53  -0.5528236
#> T67  -0.4452930
#> T71  -0.3689358
fixef(m2n)
#> (Intercept)          ti         ti2
#>  -11.223310   12.712094   -2.913682
```

Key Takeaways

- `lme4` is preferred for general mixed models due to efficiency and simplicity.
- `nlme` is powerful for complex correlation structures and nested designs.
- Bayesian models (e.g., `MCMCglmm`) offer flexible inference under uncertainty.
- Always consider model diagnostics and random effects structure carefully.

Chapter 9

Nonlinear and Generalized Linear Mixed Models

Nonlinear Mixed Models (NLMMs) and Generalized Linear Mixed Models (GLMMs) extend traditional models by incorporating both fixed effects and random effects, allowing for greater flexibility in modeling complex data structures.

- NLMMs extend nonlinear models to include both fixed and random effects, accommodating nonlinear relationships in the data.
- GLMMs extend generalized linear models to include random effects, allowing for correlated data and non-constant variance structures.

9.1 Nonlinear Mixed Models

A general form of a nonlinear mixed model is:

$$Y_{ij} = f(\mathbf{x}_{ij}, \theta, \alpha_i) + \epsilon_{ij}$$

for the j-th response from the i-th cluster (or subject), where:

- $i = 1, \dots, n$ (number of clusters/subjects),
- $j = 1, \dots, n_i$ (number of observations per cluster),
- θ represents the fixed effects,
- α_i are the random effects for cluster i,
- $\mathbf{x}_{ij}$ are the regressors or design variables,
- $f(\cdot)$ is a nonlinear mean response function,

- ϵ_{ij} represents the residual error, often assumed to be normally distributed with mean 0.

NLMMs are particularly useful when the relationship between predictors and the response cannot be adequately captured by a linear model.

9.2 Generalized Linear Mixed Models

GLMMs extend GLMs by incorporating random effects, which allows for modeling data with hierarchical or clustered structures.

The conditional distribution of y_i given the random effects α_i is:

$$y_i \mid \alpha_i \sim \text{independent } f(y_i \mid \alpha)$$

where $f(y_i \mid \alpha)$ belongs to the exponential family of distributions:

$$f(y_i \mid \alpha) = \exp\left(\frac{y_i\theta_i - b(\theta_i)}{a(\phi)} - c(y_i, \phi)\right)$$

- θ_i is the canonical parameter,
- $a(\phi)$ is a dispersion parameter,
- $b(\theta_i)$ and $c(y_i, \phi)$ are specific functions defining the exponential family.

The conditional mean of y_i is related to θ_i by:

$$\mu_i = \frac{\partial b(\theta_i)}{\partial \theta_i}$$

Applying a link function $g(\cdot)$, we relate the mean response to both fixed and random effects:

$$\begin{aligned} E(y_i \mid \alpha) &= \mu_i \\ g(\mu_i) &= \mathbf{x}_i'\beta + \mathbf{z}_i'\alpha \end{aligned}$$

- $g(\cdot)$ is a known link function,
- $\mathbf{x}_i$ and $\mathbf{z}_i$ are design matrices for fixed and random effects, respectively,
- β represents fixed effects, and α represents random effects.

We also specify the distribution of the random effects:

$$\alpha \sim f(\alpha)$$

This distribution is often assumed to be multivariate normal (Law of large Number applies to fixed effects) but can be chosen (subjectively) based on the context.

9.3 Relationship Between NLMMs and GLMMs

NLMMs can be viewed as a special case of GLMMs when the inverse link function corresponds to a nonlinear transformation of the linear predictor:

$$\mathbf{Y}_i = \mathbf{f}(\mathbf{x}_i, \theta, \alpha_i) + \epsilon_i$$
$$\mathbf{Y}_i = g^{-1}(\mathbf{x}_i'\beta + \mathbf{z}_i'\alpha_i) + \epsilon_i$$

Here, $g^{-1}(\cdot)$ represents the inverse link function, corresponding to a nonlinear transformation of the fixed and random effects.

Note:
We can't derive the analytical formulation of the marginal distribution because nonlinear combination of normal variables is not normally distributed, even in the case of additive error (ϵ_i) and random effects (α_i) are both normal.

9.4 Marginal Properties of GLMMs

9.4.1 Marginal Mean of y_i

The marginal mean is obtained by integrating over the distribution of the random effects:

$$E(y_i) = E_\alpha(E(y_i \mid \alpha)) = E_\alpha(\mu_i) = E\left(g^{-1}(\mathbf{x}_i'\beta + \mathbf{z}_i'\alpha)\right)$$

Since $g^{-1}(\cdot)$ is nonlinear, this expectation cannot be simplified further without specific distributional assumptions.

9.4.1.1 Special Case: Log Link Function

For a log-link function, $g(\mu) = \log(\mu)$, the inverse link is the exponential function:

$$E(y_i) = E\left(\exp(\mathbf{x}_i'\beta + \mathbf{z}_i'\alpha)\right)$$

Using properties of the moment-generating function (MGF):

$$E(y_i) = \exp(\mathbf{x}_i'\beta) \cdot E\left(\exp(\mathbf{z}_i'\alpha)\right)$$

Here, $E(\exp(\mathbf{z}_i'\alpha))$ is the MGF of α evaluated at $\mathbf{z}_i$.

9.4.2 Marginal Variance of y_i

The variance decomposition formula applies:

$$\begin{aligned}\mathrm{Var}(y_i) &= \mathrm{Var}_\alpha\left(E(y_i \mid \alpha)\right) + E_\alpha\left(\mathrm{Var}(y_i \mid \alpha)\right) \\ &= \mathrm{Var}(\mu_i) + E\left(a(\phi)V(\mu_i)\right)\end{aligned}$$

Expressed explicitly:

$$\mathrm{Var}(y_i) = \mathrm{Var}\left(g^{-1}(\mathbf{x}_i'\beta + \mathbf{z}_i'\alpha)\right) + E\left(a(\phi)V\left(g^{-1}(\mathbf{x}_i'\beta + \mathbf{z}_i'\alpha)\right)\right)$$

Without specific assumptions about $g(\cdot)$ and the distribution of α, this is the most general form.

9.4.3 Marginal Covariance of y

Random effects induce correlation between observations within the same cluster. The covariance between y_i and y_j is:

$$\begin{aligned}\mathrm{Cov}(y_i, y_j) &= \mathrm{Cov}_\alpha\big(E(y_i \mid \alpha), E(y_j \mid \alpha)\big) + E_\alpha\big(\mathrm{Cov}(y_i, y_j \mid \alpha)\big) \\ &= \mathrm{Cov}(\mu_i, \mu_j) + E(0) \\ &= \mathrm{Cov}\left(g^{-1}(\mathbf{x}_i'\beta + \mathbf{z}_i'\alpha), g^{-1}(\mathbf{x}_j'\beta + \mathbf{z}_j'\alpha)\right)\end{aligned}$$

The second term vanishes when y_i and y_j are conditionally independent given α. This dependency structure is a hallmark of mixed models.

Example: Repeated Measurements with a Poisson GLMM

Consider repeated count measurements for subjects:

- Let y_{ij} be the j-th count for subject i.
- Assume $y_{ij} \mid \alpha_i \sim$ independent Poisson(μ_{ij}).

The model is specified as:

$$\log(\mu_{ij}) = \mathbf{x}_{ij}'\beta + \alpha_i$$

where:

- $\alpha_i \sim$ i.i.d. $N(0, \sigma_\alpha^2)$ represents subject-specific random effects,
- This is a log-link GLMM with random intercepts for subjects.

The inclusion of α_i accounts for subject-level heterogeneity, capturing unobserved variability across individuals.

9.5 Estimation in Nonlinear and Generalized Linear Mixed Models

In Linear Mixed Models, the marginal likelihood of the observed data $\mathbf{y}$ is derived by integrating out the random effects from the hierarchical formulation:

$$f(\mathbf{y}) = \int f(\mathbf{y} \mid \alpha)\, f(\alpha)\, d\alpha$$

For LMMs, both component distributions—

- the conditional distribution $f(\mathbf{y} \mid \alpha)$, and
- the random effects distribution $f(\alpha)$—

are typically assumed to be Gaussian with linear relationships. These assumptions imply that the marginal distribution of $\mathbf{y}$ is also Gaussian, allowing the integral to be solved analytically using properties of the multivariate normal distribution.

In contrast:

- For GLMMs, the conditional distribution $f(\mathbf{y} \mid \alpha)$ belongs to the exponential family but is not Gaussian in general.
- For NLMMs, the relationship between the mean response and the random (and fixed) effects is nonlinear, complicating the integral.

In both cases, the marginal likelihood integral:

$$L(\beta; \mathbf{y}) = \int f(\mathbf{y} \mid \alpha)\, f(\alpha)\, d\alpha$$

cannot be solved analytically. Consequently, estimation requires:

- Numerical Integration
- Linearization of the Model
-

9.5.1 Estimation by Numerical Integration

The marginal likelihood for parameter estimation is given by:

$$L(\beta; \mathbf{y}) = \int f(\mathbf{y} \mid \alpha)\, f(\alpha)\, d\alpha$$

To estimate the fixed effects β, we often maximize the log-likelihood:

$$\ell(\beta) = \log L(\beta; \mathbf{y})$$

Optimization requires the score function (gradient):

$$\frac{\partial \ell(\beta)}{\partial \beta}$$

Since the integral in $L(\beta; \mathbf{y})$ is generally intractable, we rely on numerical techniques to approximate it.

9.5.1.1 Methods for Numerical Integration

1. Gaussian Quadrature

   - Suitable for low-dimensional random effects ($\dim(\alpha)$ is small).
   - Approximates the integral using weighted sums of function evaluations at specific points (nodes).
   - Gauss-Hermite quadrature is commonly used when random effects are normally distributed.

   Limitation: Computational cost grows exponentially with the dimension of α (curse of dimensionality).

2. Laplace Approximation

   - Approximates the integral by expanding the log-likelihood around the mode of the integrand (i.e., the most likely value of α).
   - Provides accurate results for moderate-sized random effects and large sample sizes.
   - First-order Laplace approximation is commonly used; higher-order versions improve accuracy but increase complexity.

   Key Idea: Approximate the integral as:

   $$\int e^{h(\alpha)} d\alpha \approx e^{h(\hat{\alpha})} \sqrt{\frac{(2\pi)^q}{|\mathbf{H}|}}$$

   where:

   - $\hat{\alpha}$ is the mode of $h(\alpha)$,
   - $\mathbf{H}$ is the Hessian matrix of second derivatives at $\hat{\alpha}$,
   - q is the dimension of α.

3. Monte Carlo Integration

- Uses random sampling to approximate the integral.
- Importance Sampling improves efficiency by sampling from a distribution that better matches the integrand.
- Markov Chain Monte Carlo methods, such as Gibbs sampling or Metropolis-Hastings, are used when the posterior distribution is complex.

Advantage: Scales better with high-dimensional random effects compared to quadrature methods.

Limitation: Computationally intensive, and variance of estimates can be high without careful tuning.

9.5.1.2 Choosing an Integration Method

Method	Dimensionality	Accuracy	Computational Cost
Gaussian Quadrature	Low-dimensional ($q \leq 3$)	High (with sufficient nodes)	High (exponential growth with q)
Laplace Approximation	Moderate-dimensional	Moderate to High	Moderate
Monte Carlo Methods	High-dimensional	Variable (depends on sample size)	High (but scalable)

- For small random effect dimensions, quadrature methods are effective.
- For moderate dimensions, Laplace approximation offers a good balance.
- For high dimensions or complex models, Monte Carlo techniques are often the method of choice.

9.5.2 Estimation by Linearization

When estimating parameters in NLMMs and GLMMs, one common and effective approach is linearization. This technique approximates the nonlinear or non-Gaussian components with linear counterparts, enabling the use of standard LMM estimation methods. Linearization not only simplifies the estimation process but also allows for leveraging well-established statistical tools and methods developed for linear models.

9.5.2.1 Concept of Linearization

The core idea is to create a linearized version of the response variable, known as the *working response* or *pseudo-response*, denoted as $\tilde{y}_i$. This pseudo-response

is designed to approximate the original nonlinear relationship in a linear form, facilitating easier estimation of model parameters. The conditional mean of this pseudo-response is expressed as:

$$E(\tilde{y}_i \mid \alpha) = \mathbf{x}_i'\beta + \mathbf{z}_i'\alpha$$

Here:

- $\mathbf{x}_i$ is the design matrix for fixed effects,
- β represents the fixed effect parameters,
- $\mathbf{z}_i$ is the design matrix for random effects,
- α denotes the random effects.

In addition to the conditional mean, it is essential to estimate the conditional variance of the pseudo-response to fully characterize the linearized model:

$$\mathrm{Var}(\tilde{y}_i \mid \alpha)$$

This variance estimation ensures that the model accounts for the inherent variability in the data, maintaining the integrity of statistical inferences.

9.5.2.2 Application of Linearization

Once linearized, the model structure closely resembles that of a linear mixed model, allowing us to apply standard estimation techniques from LMMs. These techniques include methods such as MLE and REML, which are computationally efficient and statistically robust.

The primary difference between various linearization-based methods lies in how the linearization is performed. This often involves expanding the nonlinear function $f(\mathbf{x}, \theta, \alpha)$ or the inverse link function $g^{-1}(\cdot)$. The goal is to approximate these complex functions with simpler linear expressions while retaining as much of the original model's characteristics as possible.

9.5.2.2.1 Taylor Series Expansion A widely used method for linearization is the Taylor series expansion. This approach approximates the nonlinear mean function around initial estimates of the random effects. The first-order Taylor series expansion of the nonlinear function is given by:

$$f(\mathbf{x}_{ij}, \theta, \alpha_i) \approx f(\mathbf{x}_{ij}, \theta, \hat{\alpha}_i) + \left.\frac{\partial f}{\partial \alpha_i}\right|_{\hat{\alpha}_i} (\alpha_i - \hat{\alpha}_i)$$

In this expression:

- $f(\mathbf{x}_{ij}, \theta, \hat{\alpha}_i)$ is the function evaluated at the initial estimates of the random effects $\hat{\alpha}_i$,
- $\frac{\partial f}{\partial \alpha_i}\big|_{\hat{\alpha}_i}$ represents the gradient (or derivative) of the function with respect to the random effects, evaluated at $\hat{\alpha}_i$,
- $(\alpha_i - \hat{\alpha}_i)$ captures the deviation from the initial estimates.

The initial estimates $\hat{\alpha}_i$ are often set to zero for simplicity, especially in the early stages of model fitting. This approximation reduces the model to a linear form, making it amenable to standard LMM estimation techniques.

9.5.2.2.2 Advantages and Considerations Linearization offers several advantages:

1. Simplified Computation: By transforming complex nonlinear relationships into linear forms, linearization reduces computational complexity.
2. Flexibility: Despite the simplification, linearized models retain the ability to capture key features of the original data structure.
3. Statistical Robustness: The use of established LMM estimation techniques ensures robust parameter estimation.

However, linearization also comes with considerations:

- Approximation Error: The accuracy of the linearized model depends on how well the linear approximation captures the original nonlinear relationship.
- Choice of Expansion Point: The selection of initial estimates $\hat{\alpha}_i$ can influence the quality of the approximation.
- Higher-Order Terms: In cases where the first-order approximation is insufficient, higher-order Taylor series terms may be needed, increasing model complexity.

9.5.2.3 Penalized Quasi-Likelihood

Penalized Quasi-Likelihood (PQL) is one of the most popular linearization-based estimation methods for GLMMs.

The linearization is achieved through a first-order Taylor expansion of the inverse link function around current estimates of the parameters. The working response at the k-th iteration is given by:

$$\tilde{y}_i^{(k)} = \hat{\eta}_i^{(k-1)} + \left(y_i - \hat{\mu}_i^{(k-1)}\right) \cdot \frac{d\eta}{d\mu}\bigg|_{\hat{\eta}_i^{(k-1)}}$$

Where:

- $\eta_i = g(\mu_i)$ is the linear predictor,
- $\hat{\eta}_i^{(k-1)}$ and $\hat{\mu}_i^{(k-1)}$ are the estimates from the previous iteration $(k-1)$,
- $g(\cdot)$ is the link function, and $\mu_i = g^{-1}(\eta_i)$.

PQL Estimation Algorithm

1. Initialization:
   Start with initial estimates of β and α (commonly set to zeros).
2. Compute the Working Response:
   Use the formula above to compute $\tilde{y}_i^{(k)}$ based on current parameter estimates.
3. Fit a Linear Mixed Model:
   Apply standard LMM estimation techniques to the pseudo-response $\tilde{y}_i^{(k)}$ to update estimates of β and α.
4. Update Variance Components:
   Estimate $\mathrm{Var}(\tilde{y}_i \mid \alpha)$ based on updated parameter estimates.
5. Iteration:
   Repeat steps 2–4 until the estimates converge.

Comments on PQL:

- Advantages:
  - Easy to implement using existing LMM software.
  - Fast convergence for many practical datasets.
- Limitations:
  - Inference is only asymptotically correct due to the linearization approximation.
  - Biased estimates are common, especially:
    * For binomial responses with small group sizes,
    * In Bernoulli models (worst-case scenario),
    * In Poisson models with small counts. (Faraway, 2016)
  - Hypothesis testing and confidence intervals can be unreliable.

9.5.2.4 Generalized Estimating Equations

Generalized Estimating Equations (GEE) offer an alternative approach to parameter estimation in models with correlated data, particularly for marginal models where the focus is on population-averaged effects rather than subject-specific effects.

GEE estimates are obtained by solving estimating equations rather than maximizing a likelihood function.

Consider a marginal generalized linear model:

$$\text{logit}(E(\mathbf{y})) = \mathbf{X}\beta$$

Assuming a working covariance matrix $\mathbf{V}$ for the elements of $\mathbf{y}$, the estimating equation for β is:

$$\mathbf{X}'\mathbf{V}^{-1}(\mathbf{y} - E(\mathbf{y})) = 0$$

If $\mathbf{V}$ correctly specifies the covariance structure, the estimator is unbiased. In practice, we often assume a simple structure (e.g., independence) and obtain robust standard errors even when the covariance is misspecified.

9.5.2.4.1 GEE for Repeated Measures Let y_{ij} denote the j-th measurement on the i-th subject, with:

$$\mathbf{y}_i = \begin{pmatrix} y_{i1} \\ \vdots \\ y_{in_i} \end{pmatrix}, \quad \mu_i = \begin{pmatrix} \mu_{i1} \\ \vdots \\ \mu_{in_i} \end{pmatrix}, \quad \mathbf{x}_{ij} = \begin{pmatrix} X_{ij1} \\ \vdots \\ X_{ijp} \end{pmatrix}$$

Define the working covariance matrix of $\mathbf{y}_i$ as:

$$\mathbf{V}_i = \text{Cov}(\mathbf{y}_i)$$

Following (Liang and Zeger, 1986), the GEE for estimating β is:

$$S(\beta) = \sum_{i=1}^{K} \frac{\partial \mu_i'}{\partial \beta} \mathbf{V}_i^{-1}(\mathbf{y}_i - \mu_i) = 0$$

Where:

- K is the number of subjects (or clusters),
- $\mu_i = E(\mathbf{y}_i)$,
- $\mathbf{V}_i$ is the working covariance matrix.

9.5.2.4.2 Working Correlation Structures The covariance matrix $\mathbf{V}_i$ is modeled as:

$$\mathbf{V}_i = a(\phi)\,\mathbf{B}_i^{1/2}\,\mathbf{R}(c)\,\mathbf{B}_i^{1/2}$$

- $a(\phi)$ is a dispersion parameter,
- $\mathbf{B}_i$ is a diagonal matrix with variance functions $V(\mu_{ij})$ on the diagonal,
- $\mathbf{R}(c)$ is the working correlation matrix, parameterized by c. If $\mathbf{R}(c)$ is the true correlation matrix of $\mathbf{y}_i$, then $\mathbf{V}_i$ is the true covariance matrix.

Common Working Correlation Structures:

- Independence: $\mathbf{R} = \mathbf{I}$ (simplest, but ignores correlation).
- Exchangeable: Constant correlation between all pairs within a cluster.
- Autoregressive (AR(1)): Correlation decreases with time lag.
- Unstructured: Each pair has its own correlation parameter.

9.5.2.4.3 Iterative Algorithm for GEE Estimation

1. Initialization:
   - Compute an initial estimate of β using a GLM under the independence assumption ($\mathbf{R} = \mathbf{I}$).
2. Estimate the Working Correlation Matrix:
   - Based on residuals from the initial fit, estimate $\mathbf{R}(c)$.
3. Update the Covariance Matrix:
   - Calculate $\hat{\mathbf{V}}_i$ using the updated working correlation matrix.
4. Update β:

$$\beta^{(r+1)} = \beta^{(r)} + \left(\sum_{i=1}^{K} \frac{\partial \mu_i'}{\partial \beta}\,\hat{\mathbf{V}}_i^{-1}\,\frac{\partial \mu_i}{\partial \beta}\right)^{-1} \left(\sum_{i=1}^{K} \frac{\partial \mu_i'}{\partial \beta}\,\hat{\mathbf{V}}_i^{-1}(\mathbf{y}_i - \mu_i)\right)$$

5. Iteration:
   - Repeat steps 2–4 until convergence (i.e., when changes in β are negligible).

Comments on GEE:

- Advantages:
  - Provides consistent estimates of β even if the working correlation matrix is misspecified.

  - Robust standard errors (also known as "sandwich" estimators) account for potential misspecification.
- Limitations:
  - Not a likelihood-based method, so likelihood-ratio tests are not appropriate.
  - Efficiency loss if the working correlation matrix is poorly specified.
  - Estimation of random effects is not possible—GEE focuses on marginal (population-averaged) effects.

9.5.3 Estimation by Bayesian Hierarchical Models

Bayesian methods provide a flexible framework for estimating parameters in NLMMs and GLMMs. Unlike frequentist approaches that rely on MLE or linearization techniques, Bayesian estimation fully incorporates prior information and naturally accounts for uncertainty in both parameter estimation and predictions.

In the Bayesian context, we are interested in the posterior distribution of the model parameters, given the observed data $\mathbf{y}$:

$$f(\alpha, \beta \mid \mathbf{y}) \propto f(\mathbf{y} \mid \alpha, \beta)\, f(\alpha)\, f(\beta)$$

Where:

- $f(\mathbf{y} \mid \alpha, \beta)$ is the likelihood of the data,
- $f(\alpha)$ is the prior distribution for the random effects,
- $f(\beta)$ is the prior distribution for the fixed effects,
- $f(\alpha, \beta \mid \mathbf{y})$ is the posterior distribution, which combines prior beliefs with observed data.

Advantages of Bayesian Estimation

- **No Need for Simplifying Approximations:** Bayesian methods do not require linearization or asymptotic approximations.
- **Full Uncertainty Quantification:** Provides credible intervals for parameters and predictive distributions for new data.
- **Flexible Modeling:** Easily accommodates complex hierarchical structures, non-standard distributions, and prior information.

Computational Challenges

Despite its advantages, Bayesian estimation can be computationally intensive and complex, especially for high-dimensional models. Key implementation issues include:

1. **Non-Valid Joint Distributions:**
   In some hierarchical models, specifying valid joint distributions for the

data, random effects, and parameters can be challenging.

2. **Constraints from Mean-Variance Relationships:**
   The inherent relationship between the mean and variance in GLMMs, combined with random effects, imposes constraints on the marginal covariance structure.

3. **Computational Intensity:**
   Fitting Bayesian models often requires advanced numerical techniques like Markov Chain Monte Carlo, which can be slow to converge, especially for large datasets or complex models.

9.5.3.1 Bayesian Estimation Methods

Bayesian estimation can proceed through two general approaches:

1. **Approximating the Objective Function (Marginal Likelihood)**

The marginal likelihood is typically intractable because it requires integrating over random effects:

$$f(\mathbf{y} \mid \beta) = \int f(\mathbf{y} \mid \alpha, \beta)\, f(\alpha)\, d\alpha$$

Since this integral cannot be solved analytically, we approximate it using the following methods:

- **Laplace Approximation**
  - Approximates the integral by expanding the log-likelihood around the mode of the integrand.
  - Provides an efficient, asymptotically accurate approximation when the posterior is approximately Gaussian near its mode.
- **Quadrature Methods**
  - **Gaussian quadrature** (e.g., Gauss-Hermite quadrature) is effective for low-dimensional random effects.
  - Approximates the integral by summing weighted evaluations of the function at specific points.
- **Monte Carlo Integration**
  - Uses random sampling to approximate the integral.
  - **Importance sampling** improves efficiency by drawing samples from a distribution that closely resembles the target distribution.

2. **Approximating the Model (Linearization)**

Alternatively, we can approximate the model itself using **Taylor series linearization** around current estimates of the parameters. This approach simplifies the model, making Bayesian estimation computationally more feasible, though at the cost of some approximation error.

9.5.3.2 Markov Chain Monte Carlo Methods

The most common approach for fully Bayesian estimation is **MCMC**, which generates samples from the posterior distribution through iterative simulation. Popular MCMC algorithms include:

- **Gibbs Sampling:**
  Efficient when full conditional distributions are available in closed form.
- **Metropolis-Hastings Algorithm:**
  More general and flexible, used when full conditionals are not easily sampled.
- **Hamiltonian Monte Carlo (HMC):**
  Implemented in packages like `Stan`, provides faster convergence for complex models by leveraging gradient information.

The posterior distribution is then approximated using the generated samples:

$$f(\alpha, \beta \mid \mathbf{y}) \approx \frac{1}{N} \sum_{i=1}^{N} \delta(\alpha - \alpha^{(i)}, \beta - \beta^{(i)})$$

Where N is the number of MCMC samples and $\delta(\cdot)$ is the Dirac delta function.

9.5.4 Practical Implementation in R

Several R packages facilitate Bayesian estimation for GLMMs and NLMMs:

- **GLMM Estimation:**
  - `MASS::glmmPQL` — Penalized Quasi-Likelihood for GLMMs.
  - `lme4::glmer` — Frequentist estimation for GLMMs using Laplace approximation.
  - `glmmTMB` — Handles complex random effects structures efficiently.
- **NLMM Estimation:**
  - `nlme::nlme` — Nonlinear mixed-effects modeling.
  - `lme4::nlmer` — Extends `lme4` for nonlinear mixed models.
  - `brms::brm` — Bayesian estimation via `Stan`, supporting NLMMs.
- **Bayesian Estimation:**
  - `MCMCglmm` — Implements MCMC algorithms for GLMMs.

- `brms::brm` — High-level interface for Bayesian regression models, leveraging `Stan` for efficient MCMC sampling.

Example: Non-Gaussian Repeated Measurements

Consider the case of **repeated measurements**:

- **If the data are Gaussian:** Use Linear Mixed Models.
- **If the data are non-Gaussian:** Use Nonlinear and Generalized Linear Mixed Models.

9.6 Application: Nonlinear and Generalized Linear Mixed Models

9.6.1 Binomial Data: CBPP Dataset

We will use the **CBPP dataset** from the `lme4` package to demonstrate different estimation approaches for binomial mixed models.

```
library(lme4)
data(cbpp, package = "lme4")
head(cbpp)
#>   herd incidence size period
#> 1    1         2   14      1
#> 2    1         3   12      2
#> 3    1         4    9      3
#> 4    1         0    5      4
#> 5    2         3   22      1
#> 6    2         1   18      2
```

The data contain information about contagious bovine pleuropneumonia (CBPP) cases across different herds and periods.

1. **Penalized Quasi-Likelihood**

Pros:

- Linearizes the response to create a pseudo-response, similar to linear mixed models.
- Computationally efficient.

Cons:

- Biased for binary or Poisson data with small counts.
- Random effects must be interpreted on the link scale.
- AIC/BIC values are not interpretable since PQL does not rely on full likelihood.

```r
library(MASS)
pql_cbpp <- glmmPQL(
    cbind(incidence, size - incidence) ~ period,
    random  = ~ 1 | herd,
    data    = cbpp,
    family  = binomial(link = "logit"),
    verbose = FALSE
)
summary(pql_cbpp)
#> Linear mixed-effects model fit by maximum likelihood
#>   Data: cbpp
#>   AIC BIC logLik
#>    NA  NA     NA
#>
#> Random effects:
#>  Formula: ~1 | herd
#>         (Intercept) Residual
#> StdDev:   0.5563535 1.184527
#>
#> Variance function:
#>  Structure: fixed weights
#>  Formula: ~invwt
#> Fixed effects:  cbind(incidence, size - incidence) ~ period
#>                 Value Std.Error DF   t-value p-value
#> (Intercept) -1.327364 0.2390194 38 -5.553372  0.0000
#> period2     -1.016126 0.3684079 38 -2.758156  0.0089
#> period3     -1.149984 0.3937029 38 -2.920944  0.0058
#> period4     -1.605217 0.5178388 38 -3.099839  0.0036
#>  Correlation:
#>         (Intr) perid2 perid3
#> period2 -0.399
#> period3 -0.373  0.260
#> period4 -0.282  0.196  0.182
#>
#> Standardized Within-Group Residuals:
#>        Min         Q1        Med         Q3        Max
#> -2.0591168 -0.6493095 -0.2747620  0.5170492  2.6187632
#>
#> Number of Observations: 56
#> Number of Groups: 15
```

Interpretation

```r
exp(0.556)
#> [1] 1.743684
```

The above result shows how herd-specific odds vary, accounting for random effects.

The fixed effects are interpreted similarly to logistic regression. For example, with the logit link:

- The **log odds** of having a case in **period 2** are **-1.016** less than in **period 1** (baseline).

```
summary(pql_cbpp)$tTable
#>                 Value Std.Error DF   t-value      p-value
#> (Intercept) -1.327364 0.2390194 38 -5.553372 2.333216e-06
#> period2     -1.016126 0.3684079 38 -2.758156 8.888179e-03
#> period3     -1.149984 0.3937029 38 -2.920944 5.843007e-03
#> period4     -1.605217 0.5178388 38 -3.099839 3.637000e-03
```

2. **Numerical Integration with `glmer`**

Pros:

- More accurate estimation since the method directly integrates over random effects.

Cons:

- Computationally more expensive, especially with high-dimensional random effects.
- May struggle with convergence for complex models.

```
numint_cbpp <- glmer(
    cbind(incidence, size - incidence) ~ period + (1 | herd),
    data = cbpp,
    family = binomial(link = "logit")
)
summary(numint_cbpp)
#> Generalized linear mixed model fit by maximum likelihood (Laplace
#>   Approximation) [glmerMod]
#>  Family: binomial  ( logit )
#> Formula: cbind(incidence, size - incidence) ~ period + (1 | herd)
#>    Data: cbpp
#>
#>      AIC      BIC   logLik deviance df.resid
#>    194.1    204.2    -92.0    184.1       51
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -2.3816 -0.7889 -0.2026  0.5142  2.8791
#>
#> Random effects:
```

```
#>  Groups Name        Variance Std.Dev.
#>  herd   (Intercept) 0.4123   0.6421
#> Number of obs: 56, groups:  herd, 15
#>
#> Fixed effects:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  -1.3983     0.2312  -6.048 1.47e-09 ***
#> period2      -0.9919     0.3032  -3.272 0.001068 **
#> period3      -1.1282     0.3228  -3.495 0.000474 ***
#> period4      -1.5797     0.4220  -3.743 0.000182 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Correlation of Fixed Effects:
#>         (Intr) perid2 perid3
#> period2 -0.363
#> period3 -0.340  0.280
#> period4 -0.260  0.213  0.198
```

Comparing PQL and Numerical Integration

For small datasets, the difference between PQL and numerical integration may be minimal.

```r
library(rbenchmark)
benchmark(
    "PQL (MASS)" = {
        glmmPQL(
            cbind(incidence, size - incidence) ~ period,
            random = ~ 1 | herd,
            data = cbpp,
            family = binomial(link = "logit"),
            verbose = FALSE
        )
    },
    "Numerical Integration (lme4)" = {
        glmer(
            cbind(incidence, size - incidence) ~ period + (1 | herd),
            data = cbpp,
            family = binomial(link = "logit")
        )
    },
    replications = 50,
    columns = c("test", "replications", "elapsed", "relative"),
    order = "relative"
)
```

```
#>                            test replications elapsed relative
#> 1                    PQL (MASS)           50    4.25    1.000
#> 2 Numerical Integration (lme4)           50    8.61    2.026
```

Improving Accuracy with Gauss-Hermite Quadrature

Setting `nAGQ > 1` increases the accuracy of the likelihood approximation:

```
numint_cbpp_GH <- glmer(
    cbind(incidence, size - incidence) ~ period + (1 | herd),
    data = cbpp,
    family = binomial(link = "logit"),
    nAGQ = 20
)
summary(numint_cbpp_GH)$coefficients[, 1] -
    summary(numint_cbpp)$coefficients[, 1]
#>   (Intercept)       period2       period3       period4
#> -0.0008808634  0.0005160912  0.0004066218  0.0002644629
```

3. **Bayesian Approach with `MCMCglmm`**

Pros:

- Incorporates prior information and handles complex models with intractable likelihoods.
- Provides full posterior distributions for parameters.

Cons:

- Computationally intensive, especially with large datasets or complex hierarchical structures.

```
library(MCMCglmm)
Bayes_cbpp <- MCMCglmm(
    cbind(incidence, size - incidence) ~ period,
    random = ~ herd,
    data = cbpp,
    family = "multinomial2",
    verbose = FALSE
)
summary(Bayes_cbpp)
#>
#>  Iterations = 3001:12991
#>  Thinning interval  = 10
#>  Sample size  = 1000
#>
#>  DIC: 538.7376
#>
```

```
#>  G-structure:  ~herd
#>
#>      post.mean  l-95% CI u-95% CI eff.samp
#> herd   0.03031 5.345e-17   0.1916    87.56
#>
#>  R-structure:  ~units
#>
#>       post.mean l-95% CI u-95% CI eff.samp
#> units     1.117   0.2822     2.21    299.9
#>
#>  Location effects: cbind(incidence, size - incidence) ~ period
#>
#>             post.mean l-95% CI u-95% CI eff.samp  pMCMC
#> (Intercept)   -1.5195  -2.2597  -0.8855   1000.0 <0.001 ***
#> period2       -1.2701  -2.3362  -0.2688    909.0  0.022 *
#> period3       -1.4085  -2.5627  -0.3611    755.5  0.006 **
#> period4       -1.9892  -3.3839  -0.8128    500.7 <0.001 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- `MCMCglmm` fits a residual variance component (useful with dispersion issues).

Variance Component Analysis

```
# explains less variability
apply(Bayes_cbpp$VCV, 2, sd)
#>      herd     units
#> 0.1136323 0.5339901
```

Posterior Summaries

```
summary(Bayes_cbpp)$solutions
#>             post.mean  l-95% CI   u-95% CI eff.samp pMCMC
#> (Intercept) -1.529801 -2.156645 -0.8139842 875.5644 0.001
#> period2     -1.249981 -2.385050 -0.2879657 780.4308 0.020
#> period3     -1.401787 -2.598287 -0.3815690 800.8027 0.006
#> period4     -1.936025 -3.171037 -0.7033584 432.9481 0.004
```

MCMC Diagnostics

```
library(lattice)
xyplot(as.mcmc(Bayes_cbpp$Sol), layout = c(2, 2))
```

There is no trend (i.e., well-mixed).

```
xyplot(as.mcmc(Bayes_cbpp$VCV), layout = c(2, 1))
```

For the herd variable, many of the values are 0, which suggests a problem. To address the instability in the herd effect sampling, we can either:

- Modify prior distributions,
- Increase the number of iterations

```
Bayes_cbpp2 <- MCMCglmm(
    cbind(incidence, size - incidence) ~ period,
    random = ~ herd,
    data = cbpp,
    family = "multinomial2",
    nitt = 20000,
    burnin = 10000,
    prior = list(G = list(list(V = 1, nu = 0.1))),
    verbose = FALSE
)
xyplot(as.mcmc(Bayes_cbpp2$VCV), layout = c(2, 1))
```

To change the shape of priors, in `MCMCglmm` use:

- `V` controls for the location of the distribution (default = 1)
- `nu` controls for the concentration around V (default = 0)

9.6.2 Count Data: Owl Dataset

We'll now model count data using the **Owl dataset**

```
library(glmmTMB)
library(dplyr)
data(Owls, package = "glmmTMB")
Owls <- Owls %>% rename(Ncalls = SiblingNegotiation)
```

1. **Poisson GLMM**

Modeling call counts with a Poisson distribution:

In a typical Poisson model, the Poisson mean λ is modeled as:

$$\log(\lambda) = x'\beta$$

However, if the response variable represents a rate (e.g., counts per **BroodSize**), we can model it as:

$$\log\left(\frac{\lambda}{b}\right) = x'\beta$$

This is equivalent to:

$$\log(\lambda) = \log(b) + x'\beta$$

where b represents **BroodSize**. In this formulation, we "offset" the mean by including the logarithm of b as an offset term in the model. This adjustment accounts for the varying exposure or denominator in rate-based responses.

```
owls_glmer <- glmer(
    Ncalls ~ offset(log(BroodSize)) + FoodTreatment * SexParent + (1 | Nest),
    family = poisson,
    data = Owls
)
summary(owls_glmer)
#> Generalized linear mixed model fit by maximum likelihood (Laplace
#>   Approximation) [glmerMod]
#>  Family: poisson  ( log )
#> Formula: Ncalls ~ offset(log(BroodSize)) + FoodTreatment * SexParent +
#>     (1 | Nest)
#>    Data: Owls
#>
#>      AIC      BIC   logLik deviance df.resid
#>   5212.8   5234.8  -2601.4   5202.8      594
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -3.5529 -1.7971 -0.6842  1.2689 11.4312
#>
#> Random effects:
#>  Groups Name        Variance Std.Dev.
#>  Nest   (Intercept) 0.2063   0.4542
#> Number of obs: 599, groups:  Nest, 27
#>
#> Fixed effects:
#>                                   Estimate Std. Error z value Pr(>|z|)
#> (Intercept)                        0.65585    0.09567   6.855 7.12e-12 ***
#> FoodTreatmentSatiated             -0.65612    0.05606 -11.705  < 2e-16 ***
#> SexParentMale                     -0.03705    0.04501  -0.823   0.4104
#> FoodTreatmentSatiated:SexParentMale  0.13135    0.07036   1.867   0.0619 .
```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Correlation of Fixed Effects:
#>             (Intr) FdTrtS SxPrnM
#> FdTrtmntStt -0.225
#> SexParentMl -0.292  0.491
#> FdTrtmS:SPM  0.170 -0.768 -0.605
```

- Nest explains a relatively large proportion of the variability (its standard deviation is larger than some coefficients).
- The model fit isn't great (deviance of 5202 on 594 df).

2. **Negative Binomial Model**

Addressing overdispersion using the negative binomial distribution:

```
owls_glmerNB <- glmer.nb(
    Ncalls ~ offset(log(BroodSize)) + FoodTreatment * SexParent + (1 | Nest),
    data = Owls
)
summary(owls_glmerNB)
#> Generalized linear mixed model fit by maximum likelihood (Laplace
#>   Approximation) [glmerMod]
#>  Family: Negative Binomial(0.8423)  ( log )
#> Formula: Ncalls ~ offset(log(BroodSize)) + FoodTreatment * SexParent +
#>     (1 | Nest)
#>    Data: Owls
#>
#>      AIC      BIC   logLik deviance df.resid
#>   3495.6   3522.0  -1741.8   3483.6      593
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -0.8859 -0.7737 -0.2701  0.4443  6.1432
#>
#> Random effects:
#>  Groups Name        Variance Std.Dev.
#>  Nest   (Intercept) 0.1245   0.3529
#> Number of obs: 599, groups:  Nest, 27
#>
#> Fixed effects:
#>                                    Estimate Std. Error z value Pr(>|z|)
#> (Intercept)                         0.69005    0.13400   5.149 2.61e-07 ***
#> FoodTreatmentSatiated              -0.76657    0.16509  -4.643 3.43e-06 ***
#> SexParentMale                      -0.02605    0.14575  -0.179    0.858
#> FoodTreatmentSatiated:SexParentMale  0.15680    0.20513   0.764    0.445
```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Correlation of Fixed Effects:
#>              (Intr) FdTrtS SxPrnM
#> FdTrtmntStt -0.602
#> SexParentMl -0.683  0.553
#> FdTrtmS:SPM  0.450 -0.744 -0.671
```

There is an improvement using negative binomial considering over-dispersion

```
hist(Owls$Ncalls,breaks=30)
```

3. **Zero-Inflated Model**

Handling excess zeros with a zero-inflated Poisson model:

```
library(glmmTMB)
owls_glmm <-
    glmmTMB(
        Ncalls ~ FoodTreatment * SexParent + offset(log(BroodSize)) +
            (1 | Nest),
        ziformula =  ~ 0,
        family = nbinom2(link = "log"),
        data = Owls
    )
owls_glmm_zi <-
    glmmTMB(
        Ncalls ~ FoodTreatment * SexParent + offset(log(BroodSize)) +
```

```
            (1 | Nest),
        ziformula =  ~ 1,
        family = nbinom2(link = "log"),
        data = Owls
    )

# Scale Arrival time to use as a covariate for zero-inflation parameter
Owls$ArrivalTime <- scale(Owls$ArrivalTime)
owls_glmm_zi_cov <- glmmTMB(
    Ncalls ~ FoodTreatment * SexParent +
        offset(log(BroodSize)) +
        (1 | Nest),
    ziformula =  ~ ArrivalTime,
    family = nbinom2(link = "log"),
    data = Owls
)

as.matrix(anova(owls_glmm, owls_glmm_zi))
#>              Df      AIC      BIC    logLik deviance    Chisq Chi Df
#> owls_glmm     6 3495.610 3521.981 -1741.805 3483.610       NA     NA
#> owls_glmm_zi  7 3431.646 3462.413 -1708.823 3417.646 65.96373      1
#>               Pr(>Chisq)
#> owls_glmm              NA
#> owls_glmm_zi 4.592983e-16


as.matrix(anova(owls_glmm_zi, owls_glmm_zi_cov))
#>                  Df      AIC      BIC    logLik deviance    Chisq Chi Df
#> owls_glmm_zi      7 3431.646 3462.413 -1708.823 3417.646       NA     NA
#> owls_glmm_zi_cov  8 3422.532 3457.694 -1703.266 3406.532 11.11411      1
#>                    Pr(>Chisq)
#> owls_glmm_zi               NA
#> owls_glmm_zi_cov 0.0008567362


summary(owls_glmm_zi_cov)
#>  Family: nbinom2  ( log )
#> Formula:
#> Ncalls ~ FoodTreatment * SexParent + offset(log(BroodSize)) +      (1 | Nest)
#> Zero inflation:          ~ArrivalTime
#> Data: Owls
#>
#>      AIC      BIC   logLik deviance df.resid
#>   3422.5   3457.7  -1703.3   3406.5      591
#>
```

```
#> Random effects:
#>
#> Conditional model:
#>  Groups Name        Variance Std.Dev.
#>  Nest   (Intercept) 0.07487  0.2736
#> Number of obs: 599, groups:  Nest, 27
#>
#> Dispersion parameter for nbinom2 family (): 2.22
#>
#> Conditional model:
#>                                   Estimate Std. Error z value Pr(>|z|)
#> (Intercept)                        0.84778    0.09961   8.511  < 2e-16 ***
#> FoodTreatmentSatiated             -0.39529    0.13742  -2.877  0.00402 **
#> SexParentMale                     -0.07025    0.10435  -0.673  0.50079
#> FoodTreatmentSatiated:SexParentMale  0.12388    0.16449   0.753  0.45138
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Zero-inflation model:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  -1.3018     0.1261  -10.32  < 2e-16 ***
#> ArrivalTime   0.3545     0.1074    3.30 0.000966 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`glmmTMB` can handle ZIP GLMMs since it adds automatic differentiation to existing estimation strategies.

We can see ZIP GLMM with an arrival time covariate on the zero is best.

- Arrival time has a positive effect on observing a nonzero number of calls
- Interactions are non significant, the food treatment is significant (fewer calls after eating)
- Nest variability is large in magnitude (without this, the parameter estimates change)

9.6.3 Binomial Example: Gotway Hessian Fly Data

We will analyze the **Gotway Hessian Fly** dataset from the `agridat` package to model binomial outcomes using both frequentist and Bayesian approaches.

9.6.3.1 Data Visualization

```
library(agridat)
library(ggplot2)
```

```r
library(lme4)
library(spaMM)

data(gotway.hessianfly)
dat <- gotway.hessianfly
dat$prop <- dat$y / dat$n  # Proportion of successes

ggplot(dat, aes(x = lat, y = long, fill = prop)) +
    geom_tile() +
    scale_fill_gradient(low = 'white', high = 'black') +
    geom_text(aes(label = gen, color = block)) +
    ggtitle('Gotway Hessian Fly: Proportion of Infestation')
```

9.6.3.2 Model Specification

- Fixed Effects (β): Genotype (`gen`)
- Random Effects (α): Block (`block`), accounting for spatial or experimental design variability

1. **Frequentist Approach with `glmer`**

```r
flymodel <- glmer(
    cbind(y, n - y) ~ gen + (1 | block),
    data   = dat,
    family = binomial,
    nAGQ   = 5  # Using adaptive Gauss-Hermite quadrature for accuracy
)
```

```
summary(flymodel)
#> Generalized linear mixed model fit by maximum likelihood (Adaptive
#>   Gauss-Hermite Quadrature, nAGQ = 5) [glmerMod]
#>  Family: binomial  ( logit )
#> Formula: cbind(y, n - y) ~ gen + (1 | block)
#>    Data: dat
#>
#>      AIC      BIC   logLik deviance df.resid
#>    162.2    198.9    -64.1    128.2       47
#>
#> Scaled residuals:
#>      Min       1Q   Median       3Q      Max
#> -2.38644 -1.01188  0.09631  1.03468  2.75479
#>
#> Random effects:
#>  Groups Name        Variance Std.Dev.
#>  block  (Intercept) 0.001022 0.03196
#> Number of obs: 64, groups:  block, 4
#>
#> Fixed effects:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)   1.5035     0.3914   3.841 0.000122 ***
#> genG02       -0.1939     0.5302  -0.366 0.714644
#> genG03       -0.5408     0.5103  -1.060 0.289260
#> genG04       -1.4342     0.4714  -3.043 0.002346 **
#> genG05       -0.2037     0.5429  -0.375 0.707486
#> genG06       -0.9783     0.5046  -1.939 0.052533 .
#> genG07       -0.6041     0.5111  -1.182 0.237235
#> genG08       -1.6774     0.4907  -3.418 0.000630 ***
#> genG09       -1.3984     0.4725  -2.960 0.003078 **
#> genG10       -0.6817     0.5333  -1.278 0.201181
#> genG11       -1.4630     0.4843  -3.021 0.002522 **
#> genG12       -1.4591     0.4918  -2.967 0.003010 **
#> genG13       -3.5528     0.6600  -5.383 7.31e-08 ***
#> genG14       -2.5073     0.5264  -4.763 1.90e-06 ***
#> genG15       -2.0872     0.4851  -4.302 1.69e-05 ***
#> genG16       -2.9697     0.5383  -5.517 3.46e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Interpretation:

- The **fixed effects** (`gen`) indicate how different genotypes influence the infestation probability.
- The **random effect** for `block` captures variability due to experimental

blocks, improving model robustness.

- **Odds Ratios:** Exponentiating coefficients helps interpret the impact on infestation odds.

2. **Bayesian Approach with `MCMCglmm`**

```
library(MCMCglmm)
library(coda)

Bayes_flymodel <- MCMCglmm(
    cbind(y, n - y) ~ gen,
    random  = ~ block,
    data    = dat,
    family  = "multinomial2",
    verbose = FALSE
)
summary(Bayes_flymodel)
#>
#>  Iterations = 3001:12991
#>  Thinning interval  = 10
#>  Sample size  = 1000
#>
#>  DIC: 876.5476
#>
#>  G-structure:  ~block
#>
#>       post.mean l-95% CI u-95% CI eff.samp
#> block   0.01028 6.99e-17  0.02814    872.7
#>
#>  R-structure:  ~units
#>
#>       post.mean l-95% CI u-95% CI eff.samp
#> units     1.005   0.3446      1.9    354.8
#>
#>  Location effects: cbind(y, n - y) ~ gen
#>
#>             post.mean l-95% CI u-95% CI eff.samp  pMCMC
#> (Intercept)   1.97622  0.54231  3.22591    806.8  0.004 **
#> genG02       -0.42313 -2.21175  1.42876    929.8  0.650
#> genG03       -0.73259 -2.68958  0.95494   1000.0  0.400
#> genG04       -1.84145 -3.69541 -0.21353    893.5  0.032 *
#> genG05       -0.36572 -2.09198  1.66953    878.2  0.714
#> genG06       -1.31243 -3.18999  0.43163    940.8  0.152
#> genG07       -0.74898 -2.45451  1.06109    818.6  0.414
#> genG08       -2.13599 -3.99461 -0.42532   1000.0  0.020 *
#> genG09       -1.86895 -3.61008 -0.02572    910.3  0.040 *
```

```
#> genG10       -0.82455 -2.72016  1.07709    833.3  0.336
#> genG11       -1.95852 -3.68950 -0.11623    812.8  0.024 *
#> genG12       -1.91755 -3.65886 -0.13231    857.2  0.034 *
#> genG13       -4.39311 -6.48591 -2.42346    707.8 <0.001 ***
#> genG14       -3.22574 -5.33514 -1.54308    800.6 <0.001 ***
#> genG15       -2.79275 -4.49885 -0.91989   1000.0  0.004 **
#> genG16       -3.90025 -6.10242 -2.10468    804.3  0.004 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

MCMC Diagnostics

- **Trace Plot:** Checks for chain mixing and convergence.

- **Autocorrelation Plot:** Evaluates dependency between MCMC samples.

```r
# Trace plot for the first fixed effect
plot(Bayes_flymodel$Sol[, 1],
     main = colnames(Bayes_flymodel$Sol)[1])
```

```r
# Autocorrelation plot
autocorr.plot(Bayes_flymodel$Sol[, 1],
              main = colnames(Bayes_flymodel$Sol)[1])
```

Bayesian Interpretation:

- **Posterior Means:** Represent the central tendency of the parameter estimates.
- **Credible Intervals:** Unlike frequentist confidence intervals, they can be interpreted directly as the probability that the parameter lies within the interval.

9.6.4 Nonlinear Mixed Model: Yellow Poplar Data

This dataset comes from Schabenberger and Pierce (2001)

9.6.4.1 Data Preparation

```
dat2 <- read.table("images/YellowPoplarData_r.txt")
names(dat2) <-
    c('tn', 'k', 'dbh', 'totht', 'dob', 'ht', 'maxd', 'cumv')
dat2$t <- dat2$dob / dat2$dbh
dat2$r <- 1 - dat2$dob / dat2$totht
```

9.6.4.2 Data Visualization

```
library(dplyr)

dat2 <- dat2 %>% group_by(tn) %>% mutate(
    z = case_when(
        totht < 74 ~ 'a: 0-74ft',
```

```
        totht < 88 ~ 'b: 74-88',
        totht < 95 ~ 'c: 88-95',
        totht < 99 ~ 'd: 95-99',
        totht < 104 ~ 'e: 99-104',
        totht < 109 ~ 'f: 104-109',
        totht < 115 ~ 'g: 109-115',
        totht < 120 ~ 'h: 115-120',
        totht < 140 ~ 'i: 120-150',
        TRUE ~ 'j: 150+'
    )
)

ggplot(dat2, aes(x = r, y = cumv)) +
    geom_point(size = 0.5) +
    facet_wrap(vars(z)) +
    labs(title = "Cumulative Volume vs. Relative Height by Tree Height Group")
```

9.6.4.3 Model Specification

The proposed Nonlinear Mixed Model is:

$$V_{ij} = \left(\beta_0 + (\beta_1 + b_{1i})\frac{D_i^2 H_i}{1000}\right) \exp\left[-(\beta_2 + b_{2i})t_{ij}\exp(\beta_3 t_{ij})\right] + e_{ij}$$

Where:

- b_{1i}, b_{2i} are random effects for tree i.
- e_{ij} are residual errors.

9.6.4.4 Fitting the Nonlinear Mixed Model

```
library(nlme)

tmp <- nlme(
    cumv ~ (b0 + (b1 + u1) * (dbh^2 * totht / 1000)) *
        exp(-(b2 + u2) * (t / 1000) * exp(b3 * t)),
    data = dat2,
    fixed = b0 + b1 + b2 + b3 ~ 1,
    random = pdDiag(u1 + u2 ~ 1),  # Uncorrelated random effects
    groups = ~ tn,                 # Grouping by tree
    start = list(fixed = c(b0 = 0.25, b1 = 2.3, b2 = 2.87, b3 = 6.7))
)
summary(tmp)
#> Nonlinear mixed-effects model fit by maximum likelihood
#>   Model: cumv ~ (b0 + (b1 + u1) * (dbh^2 * totht/1000)) * exp(-(b2 + u2) *      (t/1000) * exp
#>   Data: dat2
#>        AIC      BIC    logLik
#>   31103.73 31151.33 -15544.86
#>
#> Random effects:
#>  Formula: list(u1 ~ 1, u2 ~ 1)
#>  Level: tn
#>  Structure: Diagonal
#>                u1       u2 Residual
#> StdDev: 0.1508094 0.447829 2.226361
#>
#> Fixed effects:  b0 + b1 + b2 + b3 ~ 1
#>       Value  Std.Error   DF  t-value p-value
#> b0 0.249386 0.12894686 6297   1.9340  0.0532
#> b1 2.288832 0.01266804 6297 180.6777  0.0000
#> b2 2.500497 0.05606686 6297  44.5985  0.0000
#> b3 6.848871 0.02140677 6297 319.9395  0.0000
#>  Correlation:
#>    b0     b1     b2
#> b1 -0.639
#> b2  0.054  0.056
#> b3 -0.011 -0.066 -0.850
#>
#> Standardized Within-Group Residuals:
#>           Min            Q1           Med            Q3           Max
#> -6.694575e+00 -3.081861e-01 -8.907041e-05  3.469469e-01  7.855665e+00
#>
#> Number of Observations: 6636
#> Number of Groups: 336
```

```
nlme::intervals(tmp)
#> Approximate 95% confidence intervals
#>
#>  Fixed effects:
#>           lower      est.     upper
#> b0 -0.003317833 0.2493858 0.5020894
#> b1  2.264006069 2.2888323 2.3136585
#> b2  2.390620116 2.5004971 2.6103742
#> b3  6.806919325 6.8488712 6.8908232
#>
#>  Random Effects:
#>   Level: tn
#>            lower      est.     upper
#> sd(u1) 0.1376068 0.1508094 0.1652787
#> sd(u2) 0.4056207 0.4478290 0.4944295
#>
#>  Within-group standard error:
#>    lower     est.    upper
#> 2.187259 2.226361 2.266161
```

9.6.4.5 Interpretation:

- **Fixed Effects** (β): Describe the average growth pattern across all trees.
- **Random Effects** (b_i): Capture tree-specific deviations from the average trend.

This result is a bit different from the original study because of different implementation of nonlinear mixed models.

9.6.4.6 Visualizing Model Predictions

```
library(cowplot)

# Prediction function
nlmmfn <- function(fixed, rand, dbh, totht, t) {
    (fixed[1] + (fixed[2] + rand[1]) * (dbh ^ 2 * totht / 1000)) *
        exp(-(fixed[3] + rand[2]) * (t / 1000) * exp(fixed[4] * t))
}

# Function to generate plots for selected trees
plot_tree <- function(tree_id) {
    pred <- data.frame(dob = seq(1, max(dat2$dob), length.out = 100))
    pred$tn <- tree_id
    pred$dbh <- unique(dat2$dbh[dat2$tn == tree_id])
    pred$t <- pred$dob / pred$dbh
```

```
    pred$totht <- unique(dat2$totht[dat2$tn == tree_id])
    pred$r <- 1 - pred$dob / pred$totht

    pred$with_random <- predict(tmp, pred)
    pred$without_random <-
        nlmmfn(tmp$coefficients$fixed,
               c(0, 0),
               pred$dbh,
               pred$totht,
               pred$t)

    ggplot(pred) +
        geom_line(aes(x = r, y = with_random, color = 'With Random Effects')) +
        geom_line(aes(x = r, y = without_random, color = 'Without Random Effects')) +
        geom_point(data = dat2[dat2$tn == tree_id,], aes(x = r, y = cumv)) +
        labs(title = paste('Tree', tree_id), colour = "") +
        theme(legend.position = "bottom")
}

# Plotting for selected trees
p1 <- plot_tree(1)
p2 <- plot_tree(151)
p3 <- plot_tree(279)

plot_grid(p1, p2, p3)
```

- **Red Line:** Model predictions with tree-specific random effects.

- **Teal Line:** Model predictions based only on fixed effects (ignoring tree-specific variation).
- **Dots:** Observed cumulative volume for each tree.

9.7 Summary

Chapter 10

Nonparametric Regression

Nonparametric regression refers to a class of regression techniques that do not assume a specific functional form (e.g., linear, polynomial of fixed degree) for the relationship between a predictor $x \in \mathbb{R}$ (or $\mathbf{x} \in \mathbb{R}^p$) and a response variable $y \in \mathbb{R}$. Instead, nonparametric methods aim to estimate this relationship directly from the data, allowing the data to "speak for themselves."

In a standard regression framework, we have a response variable Y and one or more predictors $\mathbf{X} = (X_1, X_2, \dots, X_p)$. Let us start with a univariate setting for simplicity. We assume the following model:

$$Y = m(x) + \varepsilon,$$

where:

- $m(x) = \mathbb{E}[Y \mid X = x]$ is the **regression function** we aim to estimate,
- ε is a random **error term** (noise) with $\mathbb{E}[\varepsilon \mid X = x] = 0$ and constant variance $\mathrm{Var}(\varepsilon) = \sigma^2$.

In **parametric regression** (e.g., Linear Regression), we might assume $m(x)$ has a specific form, such as:

$$m(x) = \beta_0 + \beta_1 x + \cdots + \beta_d x^d,$$

where $\beta_0, \beta_1, \dots, \beta_d$ are parameters to be estimated. In contrast, **nonparametric regression** relaxes this assumption and employs methods that can adapt to potentially complex shapes in $m(x)$ without pre-specifying its structure.

10.1 Why Nonparametric?

10.1.1 Flexibility

Nonparametric methods can capture **nonlinear relationships** and **complex patterns** in your data more effectively than many traditional parametric methods.

- **Adaptive Fit**: They rely on the data itself to determine the shape of the relationship, rather than forcing a specific equation like $Y = \beta_0 + \beta_1 x$ (linear) or a polynomial.
- **Local Structures**: Techniques like kernel smoothing or local regression focus on small neighborhoods around each observation, allowing the model to adjust dynamically to local variations.

When This Matters:

- **Highly Variable Data**: If the data shows multiple peaks, sharp transitions, or other irregular patterns.

- **Exploratory Analysis**: When you're trying to uncover hidden structures or trends in a dataset without strong prior assumptions.

10.1.2 Fewer Assumptions

Parametric methods typically assume:

- A specific functional form (e.g., linear, quadratic).

- A specific error distribution (e.g., normal, Poisson).

Nonparametric methods, on the other hand, **relax these assumptions**, making them:

- **Robust to Misspecification**: Less risk of biased estimates due to incorrect modeling choices.
- **Flexible in Error Structure**: They can handle complex error distributions without explicitly modeling them.

When This Matters:

- **Heterogeneous Populations**: In fields like ecology, genomics, or finance, where data might come from unknown mixtures of distributions.

- **Lack of Theoretical Guidance**: If theory does not suggest a strong functional form or distribution family.

10.1.3 Interpretability

Nonparametric models can still offer valuable insights:

- **Visual Interpretations**: Methods like kernel smoothing provide smooth curves that you can plot to see how Y changes with x.
- **Tree-Based Methods**: Random forests and gradient boosting (also nonparametric in nature) can be interpreted via **variable importance** measures or **partial dependence** plots, although they can be more complex than simple curves.

While you don't get simple coefficient estimates as in Linear Regression, you can still convey how certain predictors influence the response through plots or importance metrics.

10.1.4 Practical Considerations

10.1.4.1 When to Prefer Nonparametric

1. **Larger Sample Sizes:** Nonparametric methods often need more data because they let the data "speak" rather than relying on a fixed formula.
2. **Unknown or Complex Relationships:** If you suspect strong nonlinearity or have no strong theory about the functional form, nonparametric approaches provide the flexibility to discover patterns.
3. **Exploratory or Predictive Goals:** In data-driven or machine learning contexts, minimizing predictive error often takes precedence over strict parametric assumptions.

10.1.4.2 When to Be Cautious

1. **Small Sample Sizes:** Nonparametric methods can overfit and exhibit high variance if there isn't enough data to reliably estimate the relationship.
2. **Computational Cost:** Some nonparametric methods (e.g., kernel methods, large random forests) can be computationally heavier than parametric approaches like linear regression.
3. **Strong Theoretical Models:** If domain knowledge strongly suggests a specific parametric form, ignoring that might reduce clarity or conflict with established theory.
4. **Extrapolation:** Nonparametric models typically **do not** extrapolate well beyond the observed data range, because they rely heavily on local patterns.

10.1.5 Balancing Parametric and Nonparametric Approaches

In practice, it's not always an either/or decision. Consider:

- **Semiparametric Models**: Combine parametric components (for known relationships or effects) with nonparametric components (for unknown parts).
- **Model Selection & Regularization**: Use techniques like cross-validation to choose bandwidths (kernel smoothing), number of knots (splines), or hyperparameters (tree depth) to avoid overfitting.
- **Diagnostic Tools**: Start with a simple parametric model, look at residual plots to identify patterns that might warrant a nonparametric approach.

Criterion	Parametric Methods	Nonparametric Methods
Assumptions	Requires strict assumptions (e.g., linearity, distribution form)	Minimal assumptions, flexible functional forms
Data Requirements	Often works with smaller datasets if assumptions hold	Generally more data-hungry due to flexibility
Interpretability	Straightforward coefficients, easy to explain	Visual or plot-based insights; feature importance in trees
Complexity & Overfitting	Less prone to overfitting if form is correct	Can overfit if not regularized (e.g., bandwidth selection)
Extrapolation	Can extrapolate if the assumed form is correct	Poor extrapolation outside the observed data range
Computational Cost	Typically low to moderate (e.g., $O(n)$ to $O(n^2)$) depending on method	Can be higher (e.g., repeated local estimates or ensemble methods)

Drawbacks and Challenges

1. **Curse of Dimensionality:** As the number of predictors p increases, nonparametric methods often require exponentially larger sample sizes to maintain accuracy. This phenomenon, known as the curse of dimensionality, leads to sparse data in high-dimensional spaces, making it harder to obtain reliable estimates.
2. **Choice of Hyperparameters:** Methods such as kernel smoothing and splines depend on hyperparameters like bandwidth or smoothing parameters, which must be carefully selected to balance bias and variance.
3. **Computational Complexity:** Nonparametric methods can be computationally intensive, especially with large datasets or in high-dimensional settings.

10.2 Basic Concepts in Nonparametric Estimation

10.2.1 Bias-Variance Trade-Off

For a given method of estimating $m(x)$, we denote the estimator as $\hat{m}(x)$. The **mean squared error** (MSE) at a point x is defined as:

$$\mathrm{MSE}(x) = \mathbb{E}\big[\{\hat{m}(x) - m(x)\}^2\big].$$

This MSE can be decomposed into two key components: **bias** and **variance**:

$$\mathrm{MSE}(x) = \big[\mathbb{E}[\hat{m}(x)] - m(x)\big]^2 + \mathrm{Var}(\hat{m}(x)).$$

Where:

- **Bias**: Measures the systematic error in the estimator:

$$\mathrm{Bias}^2 = \big[\mathbb{E}[\hat{m}(x)] - m(x)\big]^2.$$

- **Variance**: Measures the variability of the estimator around its expected value:

$$\mathrm{Var}(\hat{m}(x)) = \mathbb{E}\big[\{\hat{m}(x) - \mathbb{E}[\hat{m}(x)]\}^2\big].$$

Nonparametric methods often have **low bias** because they can adapt to a wide range of functions. However, this flexibility can lead to **high variance**, especially when the model captures noise rather than the underlying signal.

The **bandwidth** or **smoothing parameter** in nonparametric methods typically controls this trade-off:

- **Large bandwidth** $\Rightarrow$ smoother function $\Rightarrow$ higher bias, lower variance.
- **Small bandwidth** $\Rightarrow$ more wiggly function $\Rightarrow$ lower bias, higher variance.

Selecting an optimal bandwidth is critical, as it determines the balance between underfitting (high bias) and overfitting (high variance).

10.2.2 Kernel Smoothing and Local Averages

Many nonparametric regression estimators can be viewed as **weighted local averages** of the observed responses $\{Y_i\}$. In the univariate case, if x_i are observations of the predictor and y_i are the corresponding responses, the nonparametric estimator at a point x often takes the form:

$$\widehat{m}(x) = \sum_{i=1}^{n} w_i(x)\, y_i,$$

where the weights $w_i(x)$ depend on the distance between x_i and x, and they satisfy:

$$\sum_{i=1}^{n} w_i(x) = 1.$$

We will see how this arises more concretely in kernel regression below.

10.3 Kernel Regression

10.3.1 Basic Setup

A **kernel function** $K(\cdot)$ is a non-negative, symmetric function whose integral (or sum, in a discrete setting) equals 1. In nonparametric statistics—such as kernel density estimation or local regression—kernels serve as **weighting** mechanisms, assigning higher weights to points closer to the target location and lower weights to points farther away. Specifically, a valid kernel function must satisfy:

1. **Non-negativity**:

$$K(u) \geq 0 \quad \text{for all } u.$$

2. **Normalization**:

$$\int_{-\infty}^{\infty} K(u)\, du = 1.$$

3. **Symmetry**:

$$K(u) = K(-u) \quad \text{for all } u.$$

In practice, the **bandwidth** (sometimes called the smoothing parameter) used alongside a kernel usually has a greater impact on the quality of the estimate than the particular form of the kernel. However, choosing a suitable kernel can still influence computational efficiency and the smoothness of the resulting estimates.

10.3.1.1 Common Kernel Functions

A kernel function essentially measures *proximity*, assigning higher weights to observations x_i that are close to the target point x, and smaller weights to those farther away.

1. Gaussian Kernel

$$K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}.$$

- **Shape**: Bell-shaped and infinite support (i.e., $K(u)$ is technically nonzero for all $u \in (-\infty, \infty)$), though values decay rapidly as $|u|$ grows.
- **Usage**: Due to its smoothness and mathematical convenience (especially in closed-form expressions and asymptotic analysis), it is the most widely used kernel in both density estimation and regression smoothing.
- **Properties**: The Gaussian kernel minimizes mean square error in many asymptotic scenarios, making it a common "default choice."

2. Epanechnikov Kernel

$$K(u) = \begin{cases} \frac{3}{4}(1-u^2) & \text{if } |u| \leq 1, \\ 0 & \text{otherwise.} \end{cases}$$

- **Shape**: Parabolic (inverted) on $[-1, 1]$, dropping to 0 at $|u| = 1$.
- **Usage**: Known for being **optimal** in a minimax sense for certain classes of problems, and it is frequently preferred when compact support (zero weights outside $|u| \leq 1$) is desirable.
- **Efficiency**: Because it is only supported on a finite interval, computations often involve fewer points (those outside $|u| \leq 1$ have zero weight), which can be computationally more efficient in large datasets.

3. Uniform (or Rectangular) Kernel

$$K(u) = \begin{cases} \frac{1}{2} & \text{if } |u| \leq 1, \\ 0 & \text{otherwise.} \end{cases}$$

- **Shape**: A simple "flat top" distribution on $[-1, 1]$.
- **Usage**: Sometimes used for its simplicity. In certain methods (e.g., a "moving average" approach), the uniform kernel equates to giving all points within a fixed window the same weight.
- **Drawback**: Lacks smoothness at the boundaries u =1|u|=1 u =1, and it can introduce sharper transitions in estimates compared to smoother kernels.

4. Triangular Kernel

$$K(u) = \begin{cases} 1 - |u| & \text{if } |u| \leq 1, \\ 0 & \text{otherwise.} \end{cases}$$

- **Shape**: Forms a triangle with a peak at $u = 0$ and linearly descends to 0 at $|u| = 1$.
- **Usage**: Provides a continuous but piecewise-linear alternative to the uniform kernel; places relatively more weight near the center compared to the uniform kernel.

5. Biweight (or Quartic) Kernel

$$K(u) = \begin{cases} \frac{15}{16}\left(1 - u^2\right)^2 & \text{if } |u| \leq 1, \\ 0 & \text{otherwise.} \end{cases}$$

- **Shape**: Smooth and "bump-shaped," similar to the Epanechnikov but with a steeper drop-off near $|u| = 1$.
- **Usage**: Popular when a smoother, polynomial-based kernel with compact support is desired.

6. Cosine Kernel

$$K(u) = \begin{cases} \frac{\pi}{4}\cos\left(\frac{\pi}{2}u\right) & \text{if } |u| \leq 1, \\ 0 & \text{otherwise.} \end{cases}$$

- **Shape**: A single "arch" of a cosine wave on the interval $[-1, 1]$.
- **Usage**: Used less frequently but can be appealing for certain smoothness criteria or specific signal processing contexts.

Below is a comparison of widely used kernel functions, their functional forms, support, and main characteristics.

Kernel	**Formula**	**Support**	**Key Characteristics**
Gaussian	$K(u) = \dfrac{1}{\sqrt{2\pi}}\, e^{-\frac{u^2}{2}}$	$u \in (-\infty, \infty)$	**Smooth, bell-shaped** Nonzero for all u, but decays quickly Often the default choice due to favorable analytical properties

Kernel	Formula	Support	Key Characteristics
Epanechnikov	$K(u) = \begin{cases} \frac{3}{4}(1-u^2) & \lvert u\rvert \le 1 \\ 0 & \text{otherwise} \end{cases}$	$[-1, 1]$	**Parabolic shape** Compact support Minimizes mean integrated squared error in certain theoretical contexts
Uniform	$K(u) = \begin{cases} \frac{1}{2} & \lvert u\rvert \le 1 \\ 0 & \text{otherwise} \end{cases}$	$[-1, 1]$	**Flat (rectangular) shape** Equal weight for all points within $[-1, 1]$ Sharp boundary can lead to less smooth estimates
Triangular	$K(u) = \begin{cases} 1-\lvert u\rvert & \lvert u\rvert \le 1 \\ 0 & \text{otherwise} \end{cases}$	$[-1, 1]$	**Linear decrease** from the center $u = 0$ to 0 at $\lvert u\rvert = 1$ Compact support A bit smoother than the uniform kernel
Biweight (Quartic)	$K(u) = \begin{cases} \frac{15}{16}(1-u^2)^2 & \lvert u\rvert \le 1 \\ 0 & \text{otherwise} \end{cases}$	$[-1, 1]$	**Polynomial shape**, smooth Compact support Often used for its relatively smooth taper near the boundaries
Cosine	$K(u) = \begin{cases} \frac{\pi}{4}\cos\left(\frac{\pi}{2}u\right) & \lvert u\rvert \le 1 \\ 0 & \text{otherwise} \end{cases}$	$[-1, 1]$	**Single arch** of a cosine wave Compact support Less commonly used, but still mathematically straightforward

10.3.1.2 Additional Details and Usage Notes

1. **Smoothness and Differentiability**
   - Kernels with **infinite support** (like the Gaussian) can yield very smooth estimates but require summing over (practically) all data points.
   - Kernels with **compact support** (like Epanechnikov, biweight, triangular, etc.) go to zero outside a fixed interval. This can make computations more efficient since only data within a certain range of the target point matter.
2. **Choice of Kernel vs. Choice of Bandwidth**
   - While the kernel shape does have some effect on the estimator's smoothness, **the choice of bandwidth (sometimes denoted h)** is typically more critical. If h is too large, the estimate can be excessively smooth (high bias). If h is too small, the estimate can exhibit high variance or appear "noisy."
3. **Local Weighting Principle**
   - At a target location x, a kernel function $K(\frac{x-x_i}{h})$ down-weights data points (x_i) that are farther from x. Nearer points have larger kernel values, hence exert greater influence on the local estimate.
4. **Interpretation in Density Estimation**
   - In **kernel density estimation**, each data point contributes a small "bump" (shaped by the kernel) to the overall density. Summing or integrating these bumps yields a continuous estimate of the underlying density function, in contrast to discrete histograms.

10.3.2 Nadaraya-Watson Kernel Estimator

The most widely used kernel-based regression estimator is the Nadaraya-Watson estimator (Nadaraya, 1964; Watson, 1964), defined as:

$$\widehat{m}_h(x) = \frac{\sum_{i=1}^n K(\frac{x-x_i}{h})\, y_i}{\sum_{i=1}^n K(\frac{x-x_i}{h})},$$

where $h > 0$ is the **bandwidth parameter**. Intuitively, this formula computes a weighted average of the observed y_i values, with weights determined by the kernel function applied to the scaled distance between x and each x_i.

Interpretation:

- When $|x - x_i|$ is **small** (i.e., x_i is close to x), the kernel value $K(\frac{x-x_i}{h})$ is **large**, giving more weight to y_i.
- When $|x - x_i|$ is **large**, the kernel value becomes **small** (or even zero for compactly supported kernels like the Epanechnikov), reducing the influence of y_i on $\widehat{m}_h(x)$.

Thus, observations near x have a larger impact on the estimated value $\widehat{m}_h(x)$ than distant ones.

10.3.2.1 Weights Representation

We can define the **normalized weights**:

$$w_i(x) = \frac{K\left(\frac{x-x_i}{h}\right)}{\sum_{j=1}^{n} K\left(\frac{x-x_j}{h}\right)},$$

so that the estimator can be rewritten as:

$$\widehat{m}_h(x) = \sum_{i=1}^{n} w_i(x) y_i,$$

where $\sum_{i=1}^{n} w_i(x) = 1$ for any x. Notice that $0 \leq w_i(x) \leq 1$ for all i.

10.3.3 Priestley–Chao Kernel Estimator

The **Priestley–Chao kernel estimator** (Priestley and Chao, 1972) is an early kernel-based regression estimator designed to estimate the regression function $m(x)$ from observed data $\{(x_i, y_i)\}_{i=1}^{n}$. Unlike the Nadaraya–Watson estimator, which uses pointwise kernel weighting, the Priestley–Chao estimator incorporates **differences in the predictor variable** to approximate integrals more accurately.

The estimator is defined as:

$$\widehat{m}_h(x) = \frac{1}{h} \sum_{i=1}^{n-1} K\Big(\frac{x - x_i}{h}\Big) \cdot (x_{i+1} - x_i) \cdot y_i,$$

where:

- $K(\cdot)$ is a kernel function,
- $h > 0$ is the **bandwidth parameter**,
- $(x_{i+1} - x_i)$ represents the **spacing between consecutive observations**.

10.3.3.1 Interpretation

- The estimator can be viewed as a **Riemann sum approximation** of an integral, where the kernel-weighted y_i values are scaled by the spacing $(x_{i+1} - x_i)$.

- Observations where x_i is **close to** x receive more weight due to the kernel function.
- The inclusion of $(x_{i+1} - x_i)$ accounts for **non-uniform spacing** in the data, making the estimator more accurate when the predictor values are irregularly spaced.

This estimator is particularly useful when the design points $\{x_i\}$ are **unevenly distributed**.

10.3.3.2 Weights Representation

We can express the estimator as a **weighted sum** of the observed responses y_i:

$$\widehat{m}_h(x) = \sum_{i=1}^{n-1} w_i(x)\, y_i,$$

where the weights are defined as:

$$w_i(x) = \frac{1}{h} \cdot K\Big(\frac{x - x_i}{h}\Big) \cdot (x_{i+1} - x_i).$$

Properties of the weights:

- **Non-negativity:** If $K(u) \geq 0$, then $w_i(x) \geq 0$.
- **Adaptation to spacing:** Larger gaps $(x_{i+1} - x_i)$ increase the corresponding weight.
- Unlike Nadaraya–Watson, the weights **do not sum to 1**, as they approximate an integral rather than a normalized average.

10.3.4 Gasser–Müller Kernel Estimator

The **Gasser–Müller kernel estimator** (Gasser and Müller, 1979) improves upon the Priestley–Chao estimator by using a **cumulative kernel function** to smooth over the predictor space. This estimator is particularly effective for **irregularly spaced data** and aims to reduce bias at the boundaries.

The estimator is defined as:

$$\widehat{m}_h(x) = \frac{1}{h} \sum_{i=1}^{n-1} \Big[K^*\Big(\frac{x - x_i}{h}\Big) - K^*\Big(\frac{x - x_{i+1}}{h}\Big)\Big] \cdot y_i,$$

where:

- $K^*(u) = \int_{-\infty}^{u} K(v)\,dv$ is the **cumulative distribution function (CDF)** of the kernel K,
- $h > 0$ is the **bandwidth parameter**.

10.3.4.1 Interpretation

- The estimator computes the **difference of cumulative kernel functions** at two consecutive design points, effectively assigning weight to the interval between x_i and x_{i+1}.
- Observations contribute more to $\widehat{m}_h(x)$ when x lies **between** x_i and x_{i+1}, with the contribution decreasing as the distance from x increases.
- This method smooths over **intervals** rather than just at points, reducing bias near the boundaries and improving performance with unevenly spaced data.

10.3.4.2 Weights Representation

The Gasser–Müller estimator can also be expressed as a **weighted sum**:

$$\widehat{m}_h(x) = \sum_{i=1}^{n-1} w_i(x)\, y_i,$$

where the weights are:

$$w_i(x) = \frac{1}{h}\left[K^*\Big(\frac{x - x_i}{h}\Big) - K^*\Big(\frac{x - x_{i+1}}{h}\Big)\right].$$

Properties of the weights:

- **Non-negativity:** The weights are non-negative if K^* is non-decreasing (which holds if K is non-negative).
- **Adaptation to spacing:** The weights account for the spacing between x_i and x_{i+1}.
- Similar to the Priestley–Chao estimator, the weights **do not sum to 1** because the estimator approximates an integral rather than a normalized sum.

10.3.5 Comparison of Kernel-Based Estimators

Estimator	Formula	Key Feature	Weights Sum to 1?
Nadaraya–Watson	$\hat{m}_h(x) = \frac{\sum K\left(\frac{x-x_i}{h}\right) y_i}{\sum K\left(\frac{x-x_i}{h}\right)}$	Weighted average of y_i	Yes
Priestley–Chao	$\hat{m}_h(x) = \frac{1}{h} \sum K\left(\frac{x-x_i}{h}\right)(x_{i+1} - x_i) y_i$	Incorporates data spacing	No
Gasser–Müller	$\hat{m}_h(x) = \frac{1}{h} \sum \left[K^*\left(\frac{x-x_i}{h}\right) - K^*\left(\frac{x-x_{i+1}}{h}\right)\right] y_i$	Uses cumulative kernel differences	No

---

10.3.6 Bandwidth Selection

The choice of **bandwidth** h is crucial because it controls the trade-off between bias and variance:

- If h is **too large**, the estimator becomes **overly smooth**, incorporating too many distant data points. This leads to **high bias** but **low variance**.
- If h is **too small**, the estimator becomes **noisy** and sensitive to fluctuations in the data, resulting in **low bias** but **high variance**.

10.3.6.1 Mean Squared Error and Optimal Bandwidth

To analyze the performance of kernel estimators, we often examine the **mean integrated squared error** (MISE):

$$\text{MISE}(\hat{m}_h) = \mathbb{E}\left[\int \{\hat{m}_h(x) - m(x)\}^2 \, dx\right].$$

As $n \to \infty$, under smoothness assumptions on $m(x)$ and regularity conditions on the kernel K, the MISE has the following asymptotic expansion:

$$\text{MISE}(\hat{m}_h) \approx \frac{R(K)}{nh} \sigma^2 + \frac{1}{4} \mu_2^2(K) \, h^4 \int \{m''(x)\}^2 \, dx,$$

where:

- $R(K) = \int_{-\infty}^{\infty} K(u)^2 du$ measures the **roughness** of the kernel.
- $\mu_2(K) = \int_{-\infty}^{\infty} u^2 K(u) du$ is the **second moment** of the kernel (related to its spread).

- σ^2 is the **variance of the noise**, assuming $\mathrm{Var}(\varepsilon \mid X = x) = \sigma^2$.
- $m''(x)$ is the **second derivative** of the true regression function $m(x)$.

To find the **asymptotically optimal bandwidth**, we differentiate the MISE with respect to h, set the derivative to zero, and solve for h:

$$h_{\text{opt}} = \left(\frac{R(K)\,\sigma^2}{\mu_2^2(K) \int \{m''(x)\}^2\,dx} \cdot \frac{1}{n} \right)^{1/5}.$$

In practice, σ^2 and $\int \{m''(x)\}^2 dx$ are unknown and must be estimated from data. A common data-driven approach is cross-validation.

10.3.6.2 Cross-Validation

The **leave-one-out cross-validation** (LOOCV) method is widely used for bandwidth selection:

1. For each $i = 1, \dots, n$, fit the kernel estimator $\widehat{m}_{h,-i}(x)$ using all data **except** the i-th observation (x_i, y_i).
2. Compute the squared prediction error for the left-out point: $(y_i - \widehat{m}_{h,-i}(x_i))^2$.
3. Average these errors across all observations:

$$\mathrm{CV}(h) = \frac{1}{n} \sum_{i=1}^{n} \left\{ y_i - \widehat{m}_{h,-i}(x_i) \right\}^2.$$

The bandwidth h that minimizes $\mathrm{CV}(h)$ is selected as the optimal bandwidth.

10.3.7 Asymptotic Properties

For the Nadaraya-Watson estimator, under regularity conditions and assuming $h \to 0$ as $n \to \infty$ (but not too fast), we have:

- **Consistency:**

$$\widehat{m}_h(x) \xrightarrow{p} m(x),$$

  meaning the estimator converges in probability to the true regression function.

- **Rate of Convergence:** The mean squared error (MSE) decreases at the rate:

$$\mathrm{MSE}(\widehat{m}_h(x)) = O\left(n^{-4/5}\right)$$

  in the one-dimensional case. This rate results from balancing the variance term ($O(1/(nh))$) and the squared bias term ($O(h^4)$).

10.3.8 Derivation of the Nadaraya-Watson Estimator

The Nadaraya-Watson estimator can be derived from a **density-based perspective**:

1. By the definition of conditional expectation:

$$m(x) = \mathbb{E}[Y \mid X = x] = \frac{\int y\, f_{X,Y}(x,y)\, dy}{f_X(x)},$$

   where $f_{X,Y}(x,y)$ is the joint density of (X,Y), and $f_X(x)$ is the marginal density of X.

2. Estimate $f_X(x)$ using a kernel density estimator:

$$\hat{f}_X(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h} K\Big(\frac{x - x_i}{h}\Big).$$

3. Estimate the joint density $f_{X,Y}(x,y)$:

$$\hat{f}_{X,Y}(x,y) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h} K\Big(\frac{x - x_i}{h}\Big)\, \delta_{y_i}(y),$$

   where $\delta_{y_i}(y)$ is the Dirac delta function (a point mass at y_i).

4. The kernel regression estimator becomes:

$$\hat{m}_h(x) = \frac{\int y\, \hat{f}_{X,Y}(x,y)\, dy}{\hat{f}_X(x)} = \frac{\sum_{i=1}^{n} K(\frac{x-x_i}{h})\, y_i}{\sum_{i=1}^{n} K(\frac{x-x_i}{h})},$$

   which is exactly the Nadaraya-Watson estimator.

```r
# Load necessary libraries
library(ggplot2)
library(gridExtra)


# 1. Simulate Data
set.seed(123)


# Generate predictor x and response y
n <- 100
x <-
    sort(runif(n, 0, 10))  # Sorted for Priestley-Chao and Gasser-Müller
true_function <-
    function(x)
```

```
        sin(x) + 0.5 * cos(2 * x)  # True regression function

# Add Gaussian noise
y <-
    true_function(x) + rnorm(n, sd = 0.3)

# Visualization of the data
ggplot(data.frame(x, y), aes(x, y)) +
    geom_point(color = "darkblue") +
    geom_line(aes(y = true_function(x)),
              color = "red",
              linetype = "dashed") +
    labs(title = "Simulated Data with True Regression Function",
         x = "x", y = "y") +
    theme_minimal()
```

```
# Gaussian Kernel Function
gaussian_kernel <- function(u) {
    (1 / sqrt(2 * pi)) * exp(-0.5 * u ^ 2)
}

# Epanechnikov Kernel Function
epanechnikov_kernel <- function(u) {
    ifelse(abs(u) <= 1, 0.75 * (1 - u ^ 2), 0)
}

# Cumulative Kernel for Gasser-Müller (CDF of Gaussian Kernel)
```

```r
gaussian_cdf_kernel <- function(u) {
    pnorm(u, mean = 0, sd = 1)
}
```

```r
# Nadaraya-Watson Estimator
nadaraya_watson <-
    function(x_eval, x, y, h, kernel = gaussian_kernel) {
        sapply(x_eval, function(x0) {
            weights <- kernel((x0 - x) / h)
            sum(weights * y) / sum(weights)
        })
    }

# Bandwidth Selection (fixed for simplicity)
h_nw <- 0.5  # Bandwidth for Nadaraya-Watson

# Apply Nadaraya-Watson Estimator
x_grid <- seq(0, 10, length.out = 200)
nw_estimate <- nadaraya_watson(x_grid, x, y, h_nw)

# Plot Nadaraya-Watson Estimate
ggplot() +
    geom_point(aes(x, y), color = "darkblue", alpha = 0.6) +
    geom_line(aes(x_grid, nw_estimate),
              color = "green",
              linewidth = 1.2) +
    geom_line(aes(x_grid, true_function(x_grid)),
              color = "red",
              linetype = "dashed") +
    labs(
        title = "Nadaraya-Watson Kernel Estimator",
        subtitle = paste("Bandwidth (h) =", h_nw),
        x = "x",
        y = "Estimated m(x)"
    ) +
    theme_minimal()
```

- The **green curve** is the Nadaraya–Watson estimate.
- The **dashed red line** is the true regression function.
- The **blue dots** are the observed noisy data.
- The estimator **smooths the data**, assigning more weight to points close to each evaluation point based on the Gaussian kernel.

```
# Priestley-Chao Estimator
priestley_chao <-
    function(x_eval, x, y, h, kernel = gaussian_kernel) {
        sapply(x_eval, function(x0) {
            weights <- kernel((x0 - x[-length(x)]) / h) * diff(x)
            sum(weights * y[-length(y)]) / h
        })
    }

# Apply Priestley-Chao Estimator
h_pc <- 0.5
pc_estimate <- priestley_chao(x_grid, x, y, h_pc)

# Plot Priestley-Chao Estimate
ggplot() +
    geom_point(aes(x, y), color = "darkblue", alpha = 0.6) +
    geom_line(aes(x_grid, pc_estimate),
              color = "orange",
              size = 1.2) +
    geom_line(aes(x_grid, true_function(x_grid)),
```

```
                color = "red",
                linetype = "dashed") +
    labs(
        title = "Priestley-Chao Kernel Estimator",
        subtitle = paste("Bandwidth (h) =", h_pc),
        x = "x",
        y = "Estimated m(x)"
    ) +
    theme_minimal()
```

- The **orange curve** is the Priestley–Chao estimate.
- This estimator incorporates the **spacing between consecutive data points** (`diff(x)`), making it more sensitive to **non-uniform data spacing**.
- It performs similarly to Nadaraya–Watson when data are evenly spaced.

```
# Gasser-Müller Estimator
gasser_mueller <-
    function(x_eval, x, y, h, cdf_kernel = gaussian_cdf_kernel) {
        sapply(x_eval, function(x0) {
            weights <-
                (cdf_kernel((x0 - x[-length(x)]) / h) - cdf_kernel((x0 - x[-1]) / h))
            sum(weights * y[-length(y)]) / h
        })
    }
```

```r
# Apply Gasser-Müller Estimator
h_gm <- 0.5
gm_estimate <- gasser_mueller(x_grid, x, y, h_gm)

# Plot Gasser-Müller Estimate
ggplot() +
    geom_point(aes(x, y), color = "darkblue", alpha = 0.6) +
    geom_line(aes(x_grid, gm_estimate),
              color = "purple",
              size = 1.2) +
    geom_line(aes(x_grid, true_function(x_grid)),
              color = "red",
              linetype = "dashed") +
    labs(
        title = "Gasser-Müller Kernel Estimator",
        subtitle = paste("Bandwidth (h) =", h_gm),
        x = "x",
        y = "Estimated m(x)"
    ) +
    theme_minimal()
```

- The **purple curve** is the Gasser–Müller estimate.
- This estimator uses **cumulative kernel functions** to handle **irregular data spacing** and **reduce boundary bias**.
- It performs well when data are unevenly distributed.

```r
# Combine all estimates for comparison
estimates_df <- data.frame(
    x = x_grid,
    Nadaraya_Watson = nw_estimate,
    Priestley_Chao = pc_estimate,
    Gasser_Mueller = gm_estimate,
    True_Function = true_function(x_grid)
)

# Plot all estimators together
ggplot() +
  geom_point(aes(x, y), color = "gray60", alpha = 0.5) +
  geom_line(aes(x, Nadaraya_Watson, color = "Nadaraya-Watson"),
            data = estimates_df, size = 1.1) +
  geom_line(aes(x, Priestley_Chao, color = "Priestley-Chao"),
            data = estimates_df, size = 1.1) +
  geom_line(aes(x, Gasser_Mueller, color = "Gasser-Müller"),
            data = estimates_df, size = 1.1) +
  geom_line(aes(x, True_Function, color = "True Function"),
            data = estimates_df, linetype = "dashed", size = 1) +
  scale_color_manual(
    name = "Estimator",
    values = c("Nadaraya-Watson" = "green",
               "Priestley-Chao"   = "orange",
               "Gasser-Müller"    = "purple",
               "True Function"    = "red")
  ) +
  labs(
    title = "Comparison of Kernel-Based Regression Estimators",
    x = "x",
    y = "Estimated m(x)"
  ) +
  theme_minimal()
```

- **All estimators** approximate the true function well when the bandwidth is appropriately chosen.
- The Nadaraya–Watson estimator is sensitive to bandwidth selection and assumes uniform data spacing.
- The Priestley–Chao estimator accounts for **data spacing**, making it more flexible with uneven data.
- The Gasser–Müller estimator reduces **boundary bias** and handles **irregular data** effectively.

```r
# Cross-validation for bandwidth selection (for Nadaraya-Watson)
cv_bandwidth <- function(h, x, y, kernel = gaussian_kernel) {
  n <- length(y)
  cv_error <- 0
  for (i in 1:n) {
    x_train <- x[-i]
    y_train <- y[-i]
    y_pred <- nadaraya_watson(x[i], x_train, y_train, h, kernel)
    cv_error <- cv_error + (y[i] - y_pred)^2
  }
  return(cv_error / n)
}

# Optimize bandwidth
bandwidths <- seq(0.1, 2, by = 0.1)
cv_errors <- sapply(bandwidths, cv_bandwidth, x = x, y = y)
```

```r
# Optimal bandwidth
optimal_h <- bandwidths[which.min(cv_errors)]
optimal_h
#> [1] 0.3

# Plot CV errors
ggplot(data.frame(bandwidths, cv_errors), aes(bandwidths, cv_errors)) +
    geom_line(color = "blue") +
    geom_point(aes(x = optimal_h, y = min(cv_errors)),
               color = "red",
               size = 3) +
    labs(title = "Cross-Validation for Bandwidth Selection",
         x = "Bandwidth (h)", y = "CV Error") +
    theme_minimal()
```

- The **red point** indicates the **optimal bandwidth** that minimizes the cross-validation error.
- Selecting the **right bandwidth** is critical, as it balances **bias and variance** in the estimator.

10.4 Local Polynomial Regression

While the Nadaraya-Watson estimator is effectively a **local constant** estimator (it approximates $m(x)$ by a constant in a small neighborhood of x), **local poly-**

nomial regression extends this idea by fitting a **local polynomial** around each point x. The advantage of local polynomials is that they can better handle boundary bias and can capture local curvature more effectively.

10.4.1 Local Polynomial Fitting

A **local polynomial regression** of degree p at point x estimates a polynomial function:

$$m_x(t) = \beta_0 + \beta_1(t-x) + \beta_2(t-x)^2 + \cdots + \beta_p(t-x)^p$$

that best fits the data $\{(x_i, y_i)\}$ within a neighborhood of x, **weighted by a kernel**. Specifically, we solve:

$$(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p) = \underset{\beta_0,\dots,\beta_p}{\arg\min} \sum_{i=1}^{n} \left[y_i - \left\{\beta_0 + \beta_1(x_i - x) + \cdots + \beta_p(x_i - x)^p\right\}\right]^2 K\left(\frac{x_i - x}{h}\right).$$

We then estimate:

$$\hat{m}(x) = \hat{\beta}_0,$$

because β_0 is the constant term of the local polynomial expansion around x, which represents the estimated value at that point.

Why center the polynomial at x rather than 0?
Centering at x ensures that the fitted polynomial provides the best approximation *around* x. This is conceptually similar to a Taylor expansion, where local approximations are most accurate near the point of expansion.

10.4.2 Mathematical Form of the Solution

Let $\mathbf{X}_x$ be the design matrix for the local polynomial expansion at point x. For a polynomial of degree p, each row i of $\mathbf{X}_x$ is:

$$(1,\ x_i - x,\ (x_i - x)^2,\ \dots,\ (x_i - x)^p).$$

Define $\mathbf{W}_x$ as the diagonal matrix with entries:

$$(\mathbf{W}_x)_{ii} = K\left(\frac{x_i - x}{h}\right),$$

representing the kernel weights. The parameter vector $\beta = (\beta_0, \beta_1, \dots, \beta_p)^T$ is estimated via weighted least squares:

$$\widehat{\beta}(x) = \left(\mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x\right)^{-1} \mathbf{X}_x^T \mathbf{W}_x \mathbf{y},$$

where $\mathbf{y} = (y_1, y_2, \dots, y_n)^T$. The local polynomial estimator of $m(x)$ is given by:

$$\widehat{m}(x) = \widehat{\beta}_0(x).$$

Alternatively, we can express this concisely using a selection vector:

$$\widehat{m}(x) = \mathbf{e}_1^T \left(\mathbf{X}_x^T \mathbf{W}_x \mathbf{X}_x\right)^{-1} \mathbf{X}_x^T \mathbf{W}_x \mathbf{y},$$

where $\mathbf{e}_1 = (1, 0, \dots, 0)^T$ picks out the intercept term.

10.4.3 Bias, Variance, and Asymptotics

Local polynomial estimators have well-characterized bias and variance properties, which depend on the polynomial degree p, the bandwidth h, and the smoothness of the true regression function $m(x)$.

10.4.3.1 Bias

- The **leading bias term** is proportional to h^{p+1} and involves the $(p+1)$-th derivative of $m(x)$:

$$\text{Bias}\left[\widehat{m}(x)\right] \approx \frac{h^{p+1}}{(p+1)!} m^{(p+1)}(x) \cdot B(K, p),$$

  where $B(K, p)$ is a constant depending on the kernel and the polynomial degree.

- For **local linear regression** ($p = 1$), the bias is of order $O(h^2)$, while for **local quadratic regression** ($p = 2$), it's of order $O(h^3)$.

10.4.3.2 Variance

- The variance is approximately:

$$\text{Var}\left[\widehat{m}(x)\right] \approx \frac{\sigma^2}{nh} \cdot V(K, p),$$

  where σ^2 is the error variance, and $V(K, p)$ is another kernel-dependent constant.

- The variance decreases with larger n and larger h, but increasing h also increases bias, illustrating the classic **bias-variance trade-off**.

10.4.3.3 Boundary Issues

One of the key advantages of local polynomial regression—especially **local linear regression**—is its ability to reduce **boundary bias**, which is a major issue for the Nadaraya-Watson estimator. This is because the linear term allows the fit to adjust for slope changes near the boundaries, where the kernel becomes asymmetric due to fewer data points on one side.

10.4.4 Special Case: Local Linear Regression

Local linear regression (often called a local polynomial fit of degree 1) is particularly popular because:

- It **mitigates boundary bias** effectively, making it superior to Nadaraya-Watson near the edges of the data.
- It remains computationally simple yet provides better performance than local-constant (degree 0) models.
- It is **robust to heteroscedasticity**, as it adapts to varying data densities.

The resulting estimator for $\widehat{m}(x)$ simplifies to:

$$\widehat{m}(x) = \frac{S_2(x)\, S_{0y}(x) \;-\; S_1(x)\, S_{1y}(x)}{S_0(x)\, S_2(x) \;-\; S_1^2(x)},$$

where

$$S_k(x) \;=\; \sum_{i=1}^{n} (x_i - x)^k\, K\Big(\tfrac{x_i - x}{h}\Big), \quad S_{ky}(x) \;=\; \sum_{i=1}^{n} (x_i - x)^k\, y_i\, K\Big(\tfrac{x_i - x}{h}\Big).$$

To see why the local linear fit helps reduce bias, consider approximating m around the point x via a first-order Taylor expansion:

$$m(t) \;\approx\; m(x) \;+\; m'(x)\,(t - x).$$

When we perform local linear regression, we solve the weighted least squares problem

$$\min_{\beta_0, \beta_1} \sum_{i=1}^{n} \Big[y_i \;-\; \{\beta_0 + \beta_1\,(x_i - x)\} \Big]^2 K\Big(\tfrac{x_i - x}{h}\Big).$$

If we assume $y_i = m(x_i) + \varepsilon_i$, then expanding $m(x_i)$ in a Taylor series around x gives:

$$m(x_i) \;=\; m(x) \;+\; m'(x)\,(x_i - x) \;+\; \tfrac{1}{2}\,m''(x)\,(x_i - x)^2 \;+\; \cdots.$$

For x_i close to x, the higher-order terms may be small, but they contribute to the bias if we truncate at the linear term.

Let us denote:

$$S_0(x) \;=\; \sum_{i=1}^{n} K\Big(\tfrac{x_i - x}{h}\Big), \quad S_1(x) \;=\; \sum_{i=1}^{n} (x_i - x)\,K\Big(\tfrac{x_i - x}{h}\Big), \quad S_2(x) \;=\; \sum_{i=1}^{n} (x_i - x)^2\,K\Big(\tfrac{x_i - x}{h}\Big).$$

Similarly, define

$$\sum_{i=1}^{n} y_i\,K(\tfrac{x_i - x}{h}) \quad \text{and} \quad \sum_{i=1}^{n} (x_i - x)\,y_i\,K(\tfrac{x_i - x}{h})$$

for the right-hand sides. The estimated coefficients $\hat{\beta}_0$ and $\hat{\beta}_1$ are found by solving:

$$\begin{pmatrix} S_0(x) & S_1(x) \\ S_1(x) & S_2(x) \end{pmatrix} \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} y_i\,K(\frac{x_i - x}{h}) \\ \sum_{i=1}^{n} (x_i - x)\,y_i\,K(\frac{x_i - x}{h}) \end{pmatrix}.$$

Once $\hat{\beta}_0$ and $\hat{\beta}_1$ are found, we identify $\hat{m}(x) = \hat{\beta}_0$.

By substituting the Taylor expansion $y_i = m(x_i) + \varepsilon_i$ and taking expectations, one can derive how the "extra" $\frac{1}{2}\,m''(x)\,(x_i - x)^2$ terms feed into the local fit's bias.

From these expansions and associated algebra, one finds:

- **Bias:** The leading bias term for local linear regression is typically on the order of h^2, often written as $\frac{1}{2}\,m''(x)\,\mu_2(K)\,h^2$ for some constant $\mu_2(K)$ depending on the kernel's second moment.
- **Variance:** The leading variance term at a single point x is on the order of $\frac{1}{n\,h}$.

Balancing these two orders of magnitude—i.e., setting $h^2 \sim \frac{1}{n\,h}$—gives $h \sim n^{-1/3}$. Consequently, the mean squared error at x then behaves like

$$\big(\hat{m}(x) - m(x)\big)^2 \;=\; O_p(n^{-2/3}).$$

While local constant (Nadaraya–Watson) and local linear estimators often have the same *interior* rate, the local linear approach can eliminate leading-order bias near the boundaries, making it preferable in many practical settings.

10.4.5 Bandwidth Selection

Just like in kernel regression, the **bandwidth** h controls the smoothness of the local polynomial estimator.

- **Small** h: Captures fine local details but increases variance (potential overfitting).
- **Large** h: Smooths out noise but may miss important local structure (potential underfitting).

10.4.5.1 Cross-Validation for Local Polynomial Regression

Bandwidth selection via **cross-validation** is also common here. The leave-one-out CV criterion is:

$$\mathrm{CV}(h) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \widehat{m}_{-i,h}(x_i) \right)^2 ,$$

where $\widehat{m}_{-i,h}(x_i)$ is the estimate at x_i obtained by leaving out the i-th observation.

Alternatively, for local linear regression, computational shortcuts (like **generalized cross-validation**) can significantly speed up bandwidth selection.

Comparison: Nadaraya-Watson vs. Local Polynomial Regression

Aspect	Nadaraya-Watson (Local Constant)	Local Polynomial Regression
Bias at boundaries	High	Reduced (especially for $p = 1$)
Flexibility	Limited (constant fit)	Captures local trends (linear/quadratic)
Complexity	Simpler	Slightly more complex (matrix operations)
Robustness to heteroscedasticity	Lower	Higher (adapts better to varying densities)

10.4.6 Asymptotic Properties Summary

- **Consistency:** $\widehat{m}(x) \xrightarrow{p} m(x)$ as $n \to \infty$, under mild conditions.
- **Rate of Convergence:** For local linear regression, the MSE converges at rate $O(n^{-4/5})$, similar to kernel regression, but with better performance at boundaries.

- **Optimal Bandwidth:** Balances bias ($O(h^{p+1})$) and variance ($O(1/(nh))$), with cross-validation as a practical selection method.

```r
# Load necessary libraries
library(ggplot2)
library(gridExtra)

# 1. Simulate Data
set.seed(123)

# Generate predictor x and response y
n <- 100
x <- sort(runif(n, 0, 10))  # Sorted for local regression
true_function <-
    function(x)
        sin(x) + 0.5 * cos(2 * x)  # True regression function
y <-
    true_function(x) + rnorm(n, sd = 0.3) # Add Gaussian noise

# Visualization of the data
ggplot(data.frame(x, y), aes(x, y)) +
    geom_point(color = "darkblue") +
    geom_line(aes(y = true_function(x)),
              color = "red",
              linetype = "dashed") +
    labs(title = "Simulated Data with True Regression Function",
         x = "x", y = "y") +
    theme_minimal()
```

```
# Gaussian Kernel Function
gaussian_kernel <- function(u) {
    (1 / sqrt(2 * pi)) * exp(-0.5 * u ^ 2)
}

# Local Polynomial Regression Function
local_polynomial_regression <-
    function(x_eval,
             x,
             y,
             h,
             p = 1,
             kernel = gaussian_kernel) {
        sapply(x_eval, function(x0) {
            # Design matrix for polynomial of degree p
            X <- sapply(0:p, function(k)
                (x - x0) ^ k)

            # Kernel weights
            W <- diag(kernel((x - x0) / h))

            # Weighted least squares estimation
            beta_hat <- solve(t(X) %*% W %*% X, t(X) %*% W %*% y)

            # Estimated value at x0 (intercept term)
            beta_hat[1]
        })
```

```r
    }

# Evaluation grid
x_grid <- seq(0, 10, length.out = 200)

# Apply Local Linear Regression (p = 1)
h_linear <- 0.8
llr_estimate <-
    local_polynomial_regression(x_grid, x, y, h = h_linear, p = 1)

# Apply Local Quadratic Regression (p = 2)
h_quadratic <- 0.8
lqr_estimate <-
    local_polynomial_regression(x_grid, x, y, h = h_quadratic, p = 2)
```

```r
# Plot Local Linear Regression
p1 <- ggplot() +
    geom_point(aes(x, y), color = "darkblue", alpha = 0.6) +
    geom_line(aes(x_grid, llr_estimate),
              color = "green",
              size = 1.2) +
    geom_line(aes(x_grid, true_function(x_grid)),
              color = "red",
              linetype = "dashed") +
    labs(
        title = "Local Linear Regression (p = 1)",
        subtitle = paste("Bandwidth (h) =", h_linear),
        x = "x",
        y = "Estimated m(x)"
    ) +
    theme_minimal()

# Plot Local Quadratic Regression
p2 <- ggplot() +
    geom_point(aes(x, y), color = "darkblue", alpha = 0.6) +
    geom_line(aes(x_grid, lqr_estimate),
              color = "orange",
              size = 1.2) +
    geom_line(aes(x_grid, true_function(x_grid)),
              color = "red",
              linetype = "dashed") +
    labs(
        title = "Local Quadratic Regression (p = 2)",
        subtitle = paste("Bandwidth (h) =", h_quadratic),
        x = "x",
```

```
        y = "Estimated m(x)"
    ) +
    theme_minimal()

# Display plots side by side
grid.arrange(p1, p2, ncol = 2)
```

- The **green curve** represents the local linear regression estimate.
- The **orange curve** represents the **local quadratic regression** estimate.
- The **dashed red line** is the true regression function.
- **Boundary effects** are better handled by local polynomial methods, especially with quadratic fits that capture curvature more effectively.

```
# Leave-One-Out Cross-Validation for Bandwidth Selection
cv_bandwidth_lp <- function(h, x, y, p = 1, kernel = gaussian_kernel) {
  n <- length(y)
  cv_error <- 0

  for (i in 1:n) {
    # Leave-one-out data
    x_train <- x[-i]
    y_train <- y[-i]

    # Predict the left-out point
    y_pred <- local_polynomial_regression(x[i], x_train, y_train, h = h, p = p, kernel = kernel)
```

```r
    # Accumulate squared error
    cv_error <- cv_error + (y[i] - y_pred)^2
  }

  return(cv_error / n)
}

# Bandwidth grid for optimization
bandwidths <- seq(0.1, 2, by = 0.1)

# Cross-validation errors for local linear regression
cv_errors_linear <- sapply(bandwidths, cv_bandwidth_lp, x = x, y = y, p = 1)

# Cross-validation errors for local quadratic regression
cv_errors_quadratic <- sapply(bandwidths, cv_bandwidth_lp, x = x, y = y, p = 2)

# Optimal bandwidths
optimal_h_linear <- bandwidths[which.min(cv_errors_linear)]
optimal_h_quadratic <- bandwidths[which.min(cv_errors_quadratic)]

# Display optimal bandwidths
optimal_h_linear
#> [1] 0.4
optimal_h_quadratic
#> [1] 0.7
```

```r
# CV Error Plot for Linear and Quadratic Fits
cv_data <- data.frame(
    Bandwidth = rep(bandwidths, 2),
    CV_Error = c(cv_errors_linear, cv_errors_quadratic),
    Degree = rep(c("Linear (p=1)", "Quadratic (p=2)"), each = length(bandwidths))
)

ggplot(cv_data, aes(x = Bandwidth, y = CV_Error, color = Degree)) +
    geom_line(size = 1) +
    geom_point(
        data = subset(
            cv_data,
            Bandwidth %in% c(optimal_h_linear, optimal_h_quadratic)
        ),
        aes(x = Bandwidth, y = CV_Error),
        color = "red",
        size = 3
    ) +
    labs(
```

```
        title = "Cross-Validation for Bandwidth Selection",
        subtitle = "Red points indicate optimal bandwidths",
        x = "Bandwidth (h)",
        y = "CV Error"
    ) +
    theme_minimal() +
    scale_color_manual(values = c("green", "orange"))
```

- The **optimal bandwidth** minimizes the cross-validation error.
- The **red points** mark the bandwidths that yield the lowest errors for linear and quadratic fits.
- **Smaller bandwidths** can overfit, while **larger bandwidths** may over-smooth.

```
# Apply Local Polynomial Regression with Optimal Bandwidths
final_llr_estimate <-
    local_polynomial_regression(x_grid, x, y, h = optimal_h_linear, p = 1)
final_lqr_estimate <-
    local_polynomial_regression(x_grid, x, y, h = optimal_h_quadratic, p = 2)

# Plot final fits
ggplot() +
    geom_point(aes(x, y), color = "gray60", alpha = 0.5) +
    geom_line(
        aes(x_grid, final_llr_estimate, color = "Linear Estimate"),
        size = 1.2,
```

```
        linetype = "solid"
    ) +
    geom_line(
        aes(x_grid, final_lqr_estimate, color = "Quadratic Estimate"),
        size = 1.2,
        linetype = "solid"
    ) +
    geom_line(
        aes(x_grid, true_function(x_grid), color = "True Function"),
        linetype = "dashed"
    ) +
    labs(
        x = "x",
        y = "Estimated m(x)",
        color = "Legend"  # Add a legend title
    ) +
    scale_color_manual(
        values = c(
            "Linear Estimate" = "green",
            "Quadratic Estimate" = "orange",
            "True Function" = "red"
        )
    ) +
    theme_minimal()
```

10.5 Smoothing Splines

A **spline** is a piecewise polynomial function that is smooth at the junction points (called **knots**). **Smoothing splines** provide a flexible nonparametric regression technique by balancing the trade-off between closely fitting the data and maintaining smoothness. This is achieved through a penalty on the function's curvature.

In the univariate case, suppose we have data $\{(x_i, y_i)\}_{i=1}^n$ with $0 \leq x_1 < x_2 < \cdots < x_n \leq 1$ (rescaling is always possible if needed). The **smoothing spline estimator** $\hat{m}(x)$ is defined as the solution to the following optimization problem:

$$\hat{m}(x) = \underset{f \in \mathcal{H}}{\arg\min} \left\{ \sum_{i=1}^n \big(y_i - f(x_i)\big)^2 + \lambda \int_0^1 \big(f''(t)\big)^2 \, dt \right\},$$

where:

- The **first term** measures the **lack of fit** (residual sum of squares).
- The **second term** is a **roughness penalty** that discourages excessive curvature in f, controlled by the **smoothing parameter** $\lambda \geq 0$.
- The space $\mathcal{H}$ denotes the set of all twice-differentiable functions on $[0, 1]$.

Special Cases:

- **When** $\lambda = 0$: No penalty is applied, and the solution interpolates the data exactly (an interpolating spline).
- **As** $\lambda \to \infty$: The penalty dominates, forcing the solution to be as smooth as possible—reducing to a linear regression (since the second derivative of a straight line is zero).

10.5.1 Properties and Form of the Smoothing Spline

A key result from spline theory is that the minimizer $\hat{m}(x)$ is a **natural cubic spline** with knots at the observed data points $\{x_1, \ldots, x_n\}$. This result holds despite the fact that we are minimizing over an infinite-dimensional space of functions.

The solution can be expressed as:

$$\hat{m}(x) = a_0 + a_1 x + \sum_{j=1}^n b_j \, (x - x_j)_+^3,$$

where:

- $(u)_+ = \max(u, 0)$ is the **positive part function** (the cubic spline basis function),
- The coefficients $\{a_0, a_1, b_1, \ldots, b_n\}$ are determined by solving a system of linear equations derived from the optimization problem.

This form implies that the spline is a cubic polynomial within each interval between data points, with smooth transitions at the knots. The smoothness conditions ensure continuity of the function and its first and second derivatives at each knot.

10.5.2 Choice of λ

The **smoothing parameter** λ plays a crucial role in controlling the trade-off between goodness-of-fit and smoothness:

- **Large** λ: Imposes a strong penalty on the roughness, leading to a smoother (potentially underfitted) function that captures broad trends.
- **Small** λ: Allows the function to closely follow the data, possibly resulting in overfitting if the data are noisy.

A common approach to selecting λ is **generalized cross-validation (GCV)**, which provides an efficient approximation to leave-one-out cross-validation:

$$\mathrm{GCV}(\lambda) = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \widehat{m}_\lambda(x_i)\right)^2}{\left[1 - \frac{\mathrm{tr}(\mathbf{S}_\lambda)}{n}\right]^2},$$

where:

- $\widehat{m}_\lambda(x_i)$ is the fitted value at x_i for a given λ,
- $\mathbf{S}_\lambda$ is the **smoothing matrix** (or **influence matrix**) such that $\hat{\mathbf{y}} = \mathbf{S}_\lambda \mathbf{y}$,
- $\mathrm{tr}(\mathbf{S}_\lambda)$ is the **effective degrees of freedom**, reflecting the model's flexibility.

The optimal λ minimizes the GCV score, balancing fit and complexity without the need to refit the model multiple times (as in traditional cross-validation).

10.5.3 Connection to Reproducing Kernel Hilbert Spaces

Smoothing splines can be viewed through the lens of **reproducing kernel Hilbert spaces (RKHS)**. The penalty term:

$$\int_0^1 \left(f''(t)\right)^2 dt$$

defines a **semi-norm** that corresponds to the squared norm of f in a particular RKHS associated with the cubic spline kernel. This interpretation reveals that smoothing splines are equivalent to solving a **regularization problem in an RKHS**, where the penalty controls the smoothness of the solution.

This connection extends naturally to more general kernel-based methods (e.g., Gaussian process regression, kernel ridge regression) and higher-dimensional spline models.

```r
# Load necessary libraries
library(ggplot2)
library(gridExtra)

# 1. Simulate Data
set.seed(123)


# Generate predictor x and response y
n <- 100
x <- sort(runif(n, 0, 10))  # Sorted for smoother visualization
true_function <-
    function(x)
        sin(x) + 0.5 * cos(2 * x)  # True regression function
y <-
    true_function(x) + rnorm(n, sd = 0.3) # Add Gaussian noise

# Visualization of the data
ggplot(data.frame(x, y), aes(x, y)) +
    geom_point(color = "darkblue", alpha = 0.6) +
    geom_line(aes(y = true_function(x)),
              color = "red",
              linetype = "dashed") +
    labs(title = "Simulated Data with True Regression Function",
         x = "x", y = "y") +
    theme_minimal()
```

```r
# Apply Smoothing Spline with Default Lambda
# (automatically selected using GCV)
spline_fit_default <- smooth.spline(x, y)

# Apply Smoothing Spline with Manual Lambda
# (via smoothing parameter 'spar')
spline_fit_smooth <- smooth.spline(x, y, spar = 0.8)  # Smoother fit
spline_fit_flexible <-
    smooth.spline(x, y, spar = 0.4)  # More flexible fit

# Create grid for prediction
x_grid               <- seq(0, 10, length.out = 200)
spline_pred_default  <- predict(spline_fit_default, x_grid)
spline_pred_smooth   <- predict(spline_fit_smooth, x_grid)
spline_pred_flexible <- predict(spline_fit_flexible, x_grid)
```

```r
# Plot Smoothing Splines with Different Smoothness Levels
ggplot() +
    geom_point(aes(x, y), color = "darkblue", alpha = 0.5) +
    geom_line(aes(x_grid, spline_pred_default$y),
              color = "green",
              size = 1.2) +
    geom_line(
        aes(x_grid, spline_pred_smooth$y),
        color = "orange",
        size = 1.2,
        linetype = "dotted"
```

```
  ) +
  geom_line(
      aes(x_grid, spline_pred_flexible$y),
      color = "purple",
      size = 1.2,
      linetype = "dashed"
  ) +
  geom_line(
      aes(x_grid, true_function(x_grid)),
      color = "red",
      linetype = "solid",
      size = 1
  ) +
  labs(
      title = "Smoothing Spline Fits",
      subtitle = "Green: GCV-selected | Orange: Smooth (spar=0.8) | Purple: Flexible (spar=0.4)
      x = "x",
      y = "Estimated m(x)"
  ) +
  theme_minimal()
```

- The green curve is the fit with the optimal λ selected automatically via GCV.
- The orange curve (with `spar = 0.8`) is smoother, capturing broad trends but missing finer details.
- The purple curve (with `spar = 0.4`) is more flexible, fitting the data closely, potentially overfitting noise.

- The red solid line represents the true regression function.

```r
# Extract Generalized Cross-Validation (GCV) Scores
spline_fit_default$cv.crit  # GCV criterion for the default fit
#> [1] 0.09698728

# Compare GCV for different spar values
spar_values <- seq(0.1, 1.5, by = 0.05)
gcv_scores <- sapply(spar_values, function(spar) {
    fit <- smooth.spline(x, y, spar = spar)
    fit$cv.crit
})

# Optimal spar corresponding to the minimum GCV score
optimal_spar <- spar_values[which.min(gcv_scores)]
optimal_spar
#> [1] 0.7
```

```r
# GCV Plot
ggplot(data.frame(spar_values, gcv_scores),
       aes(x = spar_values, y = gcv_scores)) +
    geom_line(color = "blue", size = 1) +
    geom_point(aes(x = optimal_spar, y = min(gcv_scores)),
               color = "red",
               size = 3) +
    labs(
        title = "GCV for Smoothing Parameter Selection",
        subtitle = paste("Optimal spar =", round(optimal_spar, 2)),
        x = "Smoothing Parameter (spar)",
        y = "GCV Score"
    ) +
    theme_minimal()
```

- The blue curve shows how the GCV score changes with different smoothing parameters (`spar`).
- The red point indicates the optimal smoothing parameter that minimizes the GCV score.
- Low `spar` values correspond to flexible fits (risking overfitting), while high `spar` values produce smoother fits (risking underfitting).

```
# Apply Smoothing Spline with Optimal spar
spline_fit_optimal <- smooth.spline(x, y, spar = optimal_spar)
spline_pred_optimal <- predict(spline_fit_optimal, x_grid)

# Plot Final Fit
ggplot() +
    geom_point(aes(x, y), color = "gray60", alpha = 0.5) +
    geom_line(aes(x_grid, spline_pred_optimal$y),
              color = "green",
              size = 1.5) +
    geom_line(
        aes(x_grid, true_function(x_grid)),
        color = "red",
        linetype = "dashed",
        size = 1.2
    ) +
    labs(
        title = "Smoothing Spline with Optimal Smoothing Parameter",
        subtitle = paste("Optimal spar =", round(optimal_spar, 2)),
        x = "x",
```

```
        y = "Estimated m(x)"
    ) +
    theme_minimal()
```

10.6 Confidence Intervals in Nonparametric Regression

Constructing **confidence intervals** (or bands) for nonparametric regression estimators like kernel smoothers, local polynomials, and smoothing splines is more complex than in parametric models. The key challenges arise from the flexible nature of the models and the dependence of bias and variance on the local data density and smoothing parameters.

10.6.1 Asymptotic Normality

Under regularity conditions, nonparametric estimators are **asymptotically normal**. For a given point x, we have:

$$\sqrt{nh}\left\{\hat{m}(x) - m(x)\right\} \xrightarrow{\mathcal{D}} \mathcal{N}\left(0, \sigma^2\, \nu(x)\right),$$

where:

- n is the sample size,

- h is the bandwidth (for kernel or local polynomial estimators) or a function of λ (for smoothing splines),
- σ^2 is the variance of the errors,
- $\nu(x)$ is a function that depends on the estimator, kernel, and local data density.

An approximate $(1-\alpha)$ pointwise confidence interval for $m(x)$ is given by:

$$\widehat{m}(x) \pm z_{\alpha/2} \cdot \sqrt{\widehat{\mathrm{Var}}[\widehat{m}(x)]},$$

where:

- $z_{\alpha/2}$ is the $(1-\alpha/2)$ quantile of the standard normal distribution,
- $\widehat{\mathrm{Var}}[\widehat{m}(x)]$ is an estimate of the variance, which can be obtained using plug-in methods, sandwich estimators, or resampling techniques.

10.6.2 Bootstrap Methods

The **bootstrap** provides a powerful alternative for constructing confidence intervals and bands, particularly when asymptotic approximations are unreliable (e.g., small sample sizes or near boundaries).

10.6.2.1 Bootstrap Approaches

- **Residual Bootstrap:**
  1. Fit the nonparametric model to obtain residuals $\widehat{\varepsilon}_i = y_i - \widehat{m}(x_i)$.
  2. Generate bootstrap samples $y_i^* = \widehat{m}(x_i) + \varepsilon_i^*$, where ε_i^* are resampled (with replacement) from $\{\widehat{\varepsilon}_i\}$.
  3. Refit the model to each bootstrap sample to obtain $\widehat{m}^*(x)$.
  4. Repeat many times to build an empirical distribution of $\widehat{m}^*(x)$.
- **Wild Bootstrap:**
  Particularly useful for heteroscedastic data. Instead of resampling residuals directly, we multiply them by random variables with mean zero and unit variance to preserve the variance structure.

10.6.2.2 Bootstrap Confidence Bands

While **pointwise confidence intervals** cover the true function at a specific x with probability $(1-\alpha)$, **simultaneous confidence bands** cover the entire function over an interval with the desired confidence level. Bootstrap methods can be adapted to estimate these bands by capturing the distribution of the **maximum deviation** between $\widehat{m}(x)$ and $m(x)$ over the range of x.

10.6.3 Practical Considerations

- **Bias Correction:**
  Nonparametric estimators often have non-negligible bias, especially near

boundaries. Bias correction techniques or undersmoothing (choosing a smaller bandwidth) are sometimes used to improve interval coverage.

- **Effective Degrees of Freedom:**
  For smoothing splines, the **effective degrees of freedom** (related to $\text{tr}(\mathbf{S}_\lambda)$) provide insight into model complexity and influence confidence interval construction.

10.7 Generalized Additive Models

A generalized additive model (GAM) extends generalized linear models by allowing **additive smooth terms**:

$$g(\mathbb{E}[Y]) = \beta_0 + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p),$$

where:

- $g(\cdot)$ is a **link function** (as in GLMs),
- β_0 is the intercept,
- Each f_j is a smooth, potentially nonparametric function (e.g., a spline, kernel smoother, or local polynomial smoother),
- p is the number of predictors, with $p \geq 2$ highlighting the flexibility of GAMs in handling multivariate data.

This structure allows for **nonlinear relationships** between each predictor X_j and the response Y, while maintaining **additivity**, which simplifies interpretation compared to fully nonparametric models.

Traditional linear models assume a strictly linear relationship:

$$g(\mathbb{E}[Y]) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p.$$

However, real-world data often exhibit complex, nonlinear patterns. While generalized linear models extend linear models to non-Gaussian responses, they still rely on linear predictors. GAMs address this by replacing linear terms with smooth functions:

- **GLMs:** Linear effects (e.g., $\beta_1 X_1$)
- **GAMs:** Nonlinear smooth effects (e.g., $f_1(X_1)$)

The general form of a GAM is:

$$g(\mathbb{E}[Y \mid \mathbf{X}]) = \beta_0 + \sum_{j=1}^{p} f_j(X_j),$$

where:

- Y is the response variable,
- $\mathbf{X} = (X_1, X_2, \dots, X_p)$ are the predictors,
- f_j are smooth functions capturing potentially nonlinear effects,
- The link function $g(\cdot)$ connects the mean of Y to the additive predictor.

Special Cases:

- **When** g is the identity function and Y is continuous: This reduces to an **additive model** (a special case of GAM).
- **When** g is the logit function and Y is binary: We have a **logistic GAM** for classification tasks.
- **When** g is the log function and Y follows a Poisson distribution: This is a **Poisson GAM** for count data.

10.7.1 Estimation via Penalized Likelihood

GAMs are typically estimated using **penalized likelihood methods** to balance model fit and smoothness. The objective function is:

$$\mathcal{L}_{\text{pen}} = \ell(\beta_0, f_1, \dots, f_p) - \frac{1}{2} \sum_{j=1}^{p} \lambda_j \int \left(f_j''(x) \right)^2 dx,$$

where:

- $\ell(\beta_0, f_1, \dots, f_p)$ is the (log-)likelihood of the data,
- $\lambda_j \geq 0$ are **smoothing parameters** controlling the smoothness of each f_j,
- The penalty term $\int (f_j'')^2 dx$ discourages excessive curvature, similar to smoothing splines.

10.7.1.1 Backfitting Algorithm

For continuous responses, the classic **backfitting algorithm** is often used:

1. **Initialize**: Start with an initial guess for each f_j, typically zero.
2. **Iterate**: For each $j = 1, \dots, p$:
   - Compute the **partial residuals**:
   $$r_j = y - \beta_0 - \sum_{k \neq j} f_k(X_k)$$
   - Update f_j by fitting a smoother to (X_j, r_j).
3. **Convergence**: Repeat until the functions f_j stabilize.

This approach works because of the additive structure, which allows each smooth term to be updated conditionally on the others.

10.7.1.2 Generalized Additive Model Estimation (for GLMs)

When Y is non-Gaussian (e.g., binary, count data), we use iteratively reweighted least squares (IRLS) in combination with backfitting. Popular implementations, such as in the `mgcv` package in R, use penalized likelihood estimation with efficient computational algorithms (e.g., penalized iteratively reweighted least squares).

10.7.2 Interpretation of GAMs

One of the key advantages of GAMs is their **interpretability**, especially compared to fully nonparametric or black-box machine learning models.

- **Additive Structure:** Each predictor's effect is modeled separately via $f_j(X_j)$, making it easy to interpret marginal effects.
- **Partial Dependence Plots:** Visualization of $f_j(X_j)$ shows how each predictor affects the response, holding other variables constant.

Example:

For a marketing dataset predicting customer purchase probability:

$$\log\left(\frac{\mathbb{P}(\text{Purchase})}{1-\mathbb{P}(\text{Purchase})}\right) = \beta_0 + f_1(\text{Age}) + f_2(\text{Income}) + f_3(\text{Ad Exposure})$$

- $f_1(\text{Age})$ might show a peak in purchase likelihood for middle-aged customers.
- $f_2(\text{Income})$ could reveal a threshold effect where purchases increase beyond a certain income level.
- $f_3(\text{Ad Exposure})$ might show diminishing returns after repeated exposures.

10.7.3 Model Selection and Smoothing Parameter Estimation

The **smoothing parameters** λ_j control the complexity of each smooth term:

- **Large** λ_j: Strong smoothing, leading to nearly linear fits.
- **Small** λ_j: Flexible, wiggly fits that may overfit if λ_j is too small.

Methods for Choosing λ_j:

1. **Generalized Cross-Validation (GCV):**

$$\text{GCV} = \frac{1}{n}\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\left(1 - \frac{\text{tr}(\mathbf{S})}{n}\right)^2}$$

where $\mathbf{S}$ is the smoother matrix. GCV is a popular method for selecting the smoothing parameter λ_j because it approximates leave-one-out cross-validation without requiring explicit refitting of the model. The term $\text{tr}(\mathbf{S})$ represents the effective degrees of freedom of the smoother, and the denominator penalizes overfitting.

2. **Unbiased Risk Estimation:** This method extends the idea of GCV to non-Gaussian families (e.g., Poisson, binomial). It aims to minimize an unbiased estimate of the risk (expected prediction error). For Gaussian models, it often reduces to a form similar to GCV, but for other distributions, it incorporates the appropriate likelihood or deviance.

3. **Akaike Information Criterion (AIC):**

$$AIC = -2\log\ (L) + 2tr\ (S)$$

where L is the likelihood of the model. AIC balances model fit (measured by the likelihood) and complexity (measured by the effective degrees of freedom $tr\ (S)$). The smoothing parameter λ_j is chosen to minimize the AIC.

4. **Bayesian Information Criterion (BIC):**

$$BIC = -2\log\ (L) + \log\ (n)tr\ (S)$$

where n is the sample size. BIC is similar to AIC but imposes a stronger penalty for model complexity, making it more suitable for larger datasets.

5. **Leave-One-Out Cross-Validation (LOOCV):**

$$LOOCV = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i^{(-i)})^2,$$

where $y_i^{(-i)}$ is the predicted value for the ii-th observation when the model is fitted without it. LOOCV is computationally intensive but provides a direct estimate of prediction error.

6. **Empirical Risk Minimization:**
For some non-parametric regression methods, λ_j can be chosen by minimizing the empirical risk (e.g., mean squared error) on a validation set or via resampling techniques like k-fold cross-validation.

10.7.4 Extensions of GAMs

- **GAM with Interaction Terms:**

$$g(\mathbb{E}[Y]) = \beta_0 + f_1(X_1) + f_2(X_2) + f_{12}(X_1, X_2)$$

where f_{12} captures the interaction between X_1 and X_2 (using tensor product smooths).

- **GAMMs (Generalized Additive Mixed Models):** Incorporate random effects to handle hierarchical or grouped data.
- **Varying Coefficient Models:** Allow regression coefficients to vary smoothly with another variable, e.g.,

$$Y = \beta_0 + f_1(Z) \cdot X + \varepsilon$$

```r
# Load necessary libraries
library(mgcv)    # For fitting GAMs
library(ggplot2)
library(gridExtra)

# Simulate Data
set.seed(123)

n <- 100
x1 <- runif(n, 0, 10)
x2 <- runif(n, 0, 5)
x3 <- rnorm(n, 5, 2)

# True nonlinear functions
f1 <- function(x)
    sin(x)                  # Nonlinear effect of x1
f2 <- function(x)
    log(x + 1)              # Nonlinear effect of x2
f3 <- function(x)
    0.5 * (x - 5) ^ 2         # Quadratic effect for x3

# Generate response variable with noise
y <- 3 + f1(x1) + f2(x2) - f3(x3) + rnorm(n, sd = 1)

# Data frame for analysis
data_gam <- data.frame(y, x1, x2, x3)

# Plotting the true functions with simulated data
p1 <-
    ggplot(data_gam, aes(x1, y)) +
    geom_point() +
    labs(title = "Effect of x1 (sin(x1))")
p2 <-
    ggplot(data_gam, aes(x2, y)) +
    geom_point() +
    labs(title = "Effect of x2 (log(x2+1))")
p3 <-
```

```r
    ggplot(data_gam, aes(x3, y)) +
    geom_point() +
    labs(title = "Effect of x3 (quadratic)")

# Display plots side by side
grid.arrange(p1, p2, p3, ncol = 3)
```

```r
# Fit a GAM using mgcv
gam_model <-
    gam(y ~ s(x1) + s(x2) + s(x3),
        data = data_gam, method = "REML")

# Summary of the model
summary(gam_model)
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> y ~ s(x1) + s(x2) + s(x3)
#>
#> Parametric coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  2.63937    0.09511   27.75   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
```

```
#> Approximate significance of smooth terms:
#>          edf Ref.df       F p-value
#> s(x1) 5.997  7.165   7.966   5e-07 ***
#> s(x2) 1.000  1.000  10.249 0.00192 **
#> s(x3) 6.239  7.343 105.551 < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =   0.91   Deviance explained = 92.2%
#> -REML = 155.23  Scale est. = 0.90463   n = 100
```

```
# Plot smooth terms
par(mfrow = c(1, 3))  # Arrange plots in one row
plot(gam_model, shade = TRUE, seWithMean = TRUE)
```

```
par(mfrow = c(1, 1))  # Reset plotting layout
```

```
# Using ggplot2 with mgcv's predict function
pred_data <- with(data_gam, expand.grid(
    x1 = seq(min(x1), max(x1), length.out = 100),
    x2 = mean(x2),
    x3 = mean(x3)
))

# Predictions for x1 effect
pred_data$pred_x1 <-
    predict(gam_model, newdata = pred_data, type = "response")
```

```r
ggplot(pred_data, aes(x1, pred_x1)) +
    geom_line(color = "blue", size = 1.2) +
    labs(title = "Partial Effect of x1",
         x = "x1",
         y = "Effect on y") +
    theme_minimal()
```

```r
# Check AIC and GCV score
AIC(gam_model)
#> [1] 289.8201
gam_model$gcv.ubre  # GCV/UBRE score
#>     REML
#> 155.2314
#> attr(,"Dp")
#> [1] 47.99998


# Compare models with different smoothness
gam_model_simple <-
    gam(y ~ s(x1, k = 4) + s(x2, k = 4) + s(x3, k = 4),
        data = data_gam)
gam_model_complex <-
    gam(y ~ s(x1, k = 20) + s(x2, k = 20) + s(x3, k = 20),
        data = data_gam)

# Compare models using AIC
```

```r
AIC(gam_model, gam_model_simple, gam_model_complex)
#>                          df      AIC
#> gam_model         15.706428 289.8201
#> gam_model_simple   8.429889 322.1502
#> gam_model_complex 13.571165 287.4171
```

- Lower AIC indicates a better model balancing fit and complexity.
- GCV score helps select the optimal level of smoothness.
- Compare models to prevent overfitting (too flexible) or underfitting (too simple).

```r
# GAM with interaction using tensor product smooths
gam_interaction <- gam(y ~ te(x1, x2) + s(x3),
                       data = data_gam)

# Summary of the interaction model
summary(gam_interaction)
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> y ~ te(x1, x2) + s(x3)
#>
#> Parametric coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  2.63937    0.09364   28.19   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>             edf Ref.df       F p-value
#> te(x1,x2) 8.545  8.923   9.218  <2e-16 ***
#> s(x3)     4.766  5.834 147.595  <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.912   Deviance explained = 92.4%
#> GCV = 1.0233  Scale est. = 0.87688   n = 100

# Visualization of interaction effect
vis.gam(
    gam_interaction,
    view = c("x1", "x2"),
    plot.type = "contour",
    color = "terrain"
```

```
)
```

- The tensor product smooth `te(x1, x2)` captures nonlinear interactions between `x1` and `x2`.
- The contour plot visualizes how their joint effect influences the response.

```
# Simulate binary response
set.seed(123)
prob <- plogis(1 + f1(x1) - f2(x2) + 0.3 * x3)  # Logistic function
y_bin <- rbinom(n, 1, prob)                     # Binary outcome

# Fit GAM for binary classification
gam_logistic <-
    gam(y_bin ~ s(x1) + s(x2) + s(x3),
        family = binomial,
        data = data_gam)

# Summary and visualization
summary(gam_logistic)
#>
#> Family: binomial
#> Link function: logit
#>
#> Formula:
#> y_bin ~ s(x1) + s(x2) + s(x3)
#>
#> Parametric coefficients:
```

```
#>              Estimate Std. Error z value Pr(>|z|)
#> (Intercept)    22.30      32.18   0.693    0.488
#>
#> Approximate significance of smooth terms:
#>          edf Ref.df Chi.sq p-value
#> s(x1) 4.472  5.313  2.645   0.775
#> s(x2) 1.000  1.000  1.925   0.165
#> s(x3) 1.000  1.000  1.390   0.238
#>
#> R-sq.(adj) =      1   Deviance explained = 99.8%
#> UBRE = -0.84802  Scale est. = 1         n = 100
par(mfrow = c(1, 3))
plot(gam_logistic, shade = TRUE)
```

```
par(mfrow = c(1, 1))
```

- The logistic GAM models nonlinear effects on the log-odds of the binary outcome.
- Smooth plots indicate predictors' influence on probability of success.

```
# Diagnostic plots
par(mfrow = c(2, 2))
gam.check(gam_model)
```

```
#>
#> Method: REML   Optimizer: outer newton
#> full convergence after 9 iterations.
#> Gradient range [-5.387854e-05,2.006026e-05]
#> (score 155.2314 & scale 0.9046299).
#> Hessian positive definite, eigenvalue range [5.387409e-05,48.28647].
#> Model rank =  28 / 28
#>
#> Basis dimension (k) checking results. Low p-value (k-index<1) may
#> indicate that k is too low, especially if edf is close to k'.
#>
#>         k'  edf k-index p-value
#> s(x1) 9.00 6.00    1.01    0.46
#> s(x2) 9.00 1.00    1.16    0.92
#> s(x3) 9.00 6.24    1.07    0.72
par(mfrow = c(1, 1))
```

- Residual plots assess model fit.
- QQ plot checks for normality of residuals.
- K-index evaluates the adequacy of smoothness selection.

10.8 Regression Trees and Random Forests

Though not typically framed as "kernel" or "spline," **tree-based methods**—such as **Classification and Regression Trees (CART)** and **random**

forests—are also **nonparametric** models. They do not assume a predetermined functional form for the relationship between predictors and the response. Instead, they **adaptively partition the predictor space** into regions, fitting simple models (usually constants or linear models) within each region.

10.8.1 Regression Trees

The **Classification and Regression Trees (CART)** algorithm is the foundation of tree-based models (Breiman, 2017). In regression settings, CART models the response variable as a **piecewise constant function**.

A **regression tree** recursively partitions the predictor space into disjoint regions, $R_1, R_2, \dots, R_M$, and predicts the response as a constant within each region:

$$\hat{m}(x) = \sum_{m=1}^{M} c_m \cdot \mathbb{I}(x \in R_m),$$

where:

- c_m is the **predicted value** (usually the mean of y_i) for all observations in region R_m,
- $\mathbb{I}(\cdot)$ is the indicator function.

Tree-Building Algorithm (Greedy Recursive Partitioning):

1. **Start with the full dataset** as a single region.
2. **Find the best split**:
   - Consider all possible splits of all predictors (e.g., $X_j < s$).
   - Choose the split that minimizes the **residual sum of squares (RSS)**:
   $$\text{RSS} = \sum_{i \in R_1} (y_i - \bar{y}_{R_1})^2 + \sum_{i \in R_2} (y_i - \bar{y}_{R_2})^2,$$
   where $\bar{y}_{R_k}$ is the mean response in region R_k.
3. **Recursively repeat** the splitting process for each new region (node) until a stopping criterion is met (e.g., minimum number of observations per leaf, maximum tree depth).
4. **Assign a constant prediction** to each terminal node (leaf) based on the average response of observations within that node.

Stopping Criteria and Pruning:

- **Pre-pruning (early stopping):** Halt the tree growth when a predefined condition is met (e.g., minimal node size, maximal depth).

- **Post-pruning (cost-complexity pruning):** Grow a large tree first, then prune back to avoid overfitting. The **cost-complexity criterion** is:
$$C_\alpha(T) = \mathrm{RSS}(T) + \alpha|T|,$$
where $|T|$ is the number of terminal nodes (leaves) and α controls the penalty for complexity.

Advantages of Regression Trees:

- **Interpretability:** Easy to visualize and understand.
- **Handling of different data types:** Can naturally handle numerical and categorical variables.
- **Nonlinear relationships and interactions:** Captures complex patterns without explicit modeling.

Limitations:

- **High variance:** Trees are sensitive to small changes in data (unstable).
- **Overfitting risk:** Without pruning or regularization, deep trees can overfit noise.

10.8.2 Random Forests

To address the high variance of single trees, **random forests** combine many regression trees to create an ensemble model with improved predictive performance and stability (Breiman, 2001).

A **random forest** builds multiple decision trees and aggregates their predictions to reduce variance. For regression, the final prediction is the **average** of the predictions from individual trees:

$$\widehat{m}_{\mathrm{RF}}(x) = \frac{1}{B}\sum_{b=1}^{B} \widehat{m}^{(b)}(x),$$

where:

- B is the number of trees in the forest,
- $\widehat{m}^{(b)}(x)$ is the prediction from the b-th tree.

Random Forest Algorithm:

1. **Bootstrap Sampling:** For each tree, draw a bootstrap sample from the training data (sampling with replacement).
2. **Random Feature Selection:** At each split in the tree:
   - Randomly select a subset of predictors (usually $\sqrt{p}$ for classification or $p/3$ for regression).
   - Find the best split **only** among the selected features.

3. **Tree Growth:** Grow each tree to full depth without pruning.
4. **Aggregation:** For regression, average the predictions from all trees. For classification, use majority voting.

Why Does Random Forest Work?

- **Bagging (Bootstrap Aggregating):** Reduces variance by averaging over multiple models.
- **Random Feature Selection:** Decorrelates trees, further reducing variance.

10.8.3 Theoretical Insights

Bias-Variance Trade-off

- **Regression Trees:** Low bias but high variance.
- **Random Forests:** Slightly higher bias than a single tree (due to randomization) but significantly reduced variance, leading to lower overall prediction error.

Out-of-Bag (OOB) Error

Random forests provide an internal estimate of prediction error using **out-of-bag** samples (the data not included in the bootstrap sample for a given tree). The OOB error is computed by:

1. For each observation, predict its response using only the trees where it was **not** included in the bootstrap sample.
2. Calculate the error by comparing the OOB predictions to the true responses.

OOB error serves as an efficient, unbiased estimate of test error without the need for cross-validation.

10.8.4 Feature Importance in Random Forests

Random forests naturally provide measures of **variable importance**, helping identify which predictors contribute most to the model.

1. **Mean Decrease in Impurity (MDI):** Measures the total reduction in impurity (e.g., RSS) attributed to each variable across all trees.
2. **Permutation Importance:** Measures the increase in prediction error when the values of a predictor are randomly permuted, breaking its relationship with the response.

10.8.5 Advantages and Limitations of Tree-Based Methods

Aspect	Regression Trees	Random Forests
Interpretability	High (easy to visualize)	Moderate (difficult to interpret individual trees)
Variance	High (prone to overfitting)	Low (averaging reduces variance)
Bias	Low (flexible to data patterns)	Slightly higher than a single tree
Feature Importance	Basic (via tree splits)	Advanced (permutation-based measures)
Handling of Missing Data	Handles with surrogate splits	Handles naturally in ensemble averaging
Computational Cost	Low (fast for small datasets)	High (especially with many trees)

```r
# Load necessary libraries
library(ggplot2)
library(rpart)            # For regression trees
library(rpart.plot)        # For visualizing trees
library(randomForest)      # For random forests
library(gridExtra)

# Simulate Data
set.seed(123)

n <- 100
x1 <- runif(n, 0, 10)
x2 <- runif(n, 0, 5)
x3 <- rnorm(n, 5, 2)

# Nonlinear functions
f1 <- function(x)
    sin(x)
f2 <- function(x)
    log(x + 1)
f3 <- function(x)
    0.5 * (x - 5) ^ 2

# Generate response variable with noise
y <- 3 + f1(x1) + f2(x2) - f3(x3) + rnorm(n, sd = 1)

# Data frame
```

```r
data_tree <- data.frame(y, x1, x2, x3)

# Quick visualization of data
p1 <- ggplot(data_tree, aes(x1, y)) +
    geom_point() +
    labs(title = "Effect of x1")
p2 <- ggplot(data_tree, aes(x2, y)) +
    geom_point() +
    labs(title = "Effect of x2")
p3 <- ggplot(data_tree, aes(x3, y)) +
    geom_point() +
    labs(title = "Effect of x3")

grid.arrange(p1, p2, p3, ncol = 3)
```

```r
# Fit a Regression Tree using rpart
tree_model <-
    rpart(
        y ~ x1 + x2 + x3,
        data = data_tree,
        method = "anova",
        control = rpart.control(cp = 0.01)
    )  # cp = complexity parameter

# Summary of the tree
summary(tree_model)
#> Call:
```

```
#> rpart(formula = y ~ x1 + x2 + x3, data = data_tree, method = "anova",
#>     control = rpart.control(cp = 0.01))
#>   n= 100
#>
#>           CP nsplit rel error    xerror      xstd
#> 1 0.39895879      0 1.0000000 1.0134781 0.3406703
#> 2 0.17470339      1 0.6010412 0.8649973 0.3336272
#> 3 0.04607373      2 0.4263378 0.5707932 0.1880333
#> 4 0.02754858      3 0.3802641 0.5287366 0.1866728
#> 5 0.01584638      4 0.3527155 0.5061104 0.1867491
#> 6 0.01032524      5 0.3368691 0.5136765 0.1861020
#> 7 0.01000000      7 0.3162187 0.4847072 0.1861849
#>
#> Variable importance
#> x3 x2 x1
#> 91  6  3
#>
#> Node number 1: 100 observations,    complexity param=0.3989588
#>   mean=2.639375, MSE=9.897038
#>   left son=2 (7 obs) right son=3 (93 obs)
#>   Primary splits:
#>       x3 < 7.707736  to the right, improve=0.39895880, (0 missing)
#>       x1 < 6.84138   to the left,  improve=0.07685517, (0 missing)
#>       x2 < 2.627429  to the left,  improve=0.04029839, (0 missing)
#>
#> Node number 2: 7 observations
#>   mean=-4.603469, MSE=24.47372
#>
#> Node number 3: 93 observations,    complexity param=0.1747034
#>   mean=3.184535, MSE=4.554158
#>   left son=6 (18 obs) right son=7 (75 obs)
#>   Primary splits:
#>       x3 < 2.967495  to the left,  improve=0.40823990, (0 missing)
#>       x2 < 1.001856  to the left,  improve=0.07353453, (0 missing)
#>       x1 < 6.84138   to the left,  improve=0.07049507, (0 missing)
#>   Surrogate splits:
#>       x2 < 0.3435293 to the left,  agree=0.828, adj=0.111, (0 split)
#>
#> Node number 6: 18 observations
#>   mean=0.4012593, MSE=3.4521
#>
#> Node number 7: 75 observations,    complexity param=0.04607373
#>   mean=3.852521, MSE=2.513258
#>   left son=14 (12 obs) right son=15 (63 obs)
#>   Primary splits:
```

```
#>       x3 < 6.324486  to the right, improve=0.24191360, (0 missing)
#>       x2 < 1.603258  to the left,  improve=0.10759280, (0 missing)
#>       x1 < 6.793804  to the left,  improve=0.09106168, (0 missing)
#>
#> Node number 14: 12 observations
#>   mean=2.065917, MSE=2.252311
#>
#> Node number 15: 63 observations,    complexity param=0.02754858
#>   mean=4.192826, MSE=1.839163
#>   left son=30 (9 obs) right son=31 (54 obs)
#>   Primary splits:
#>       x3 < 3.548257  to the left,  improve=0.2353119, (0 missing)
#>       x2 < 1.349633  to the left,  improve=0.1103019, (0 missing)
#>       x1 < 7.006669  to the left,  improve=0.1019295, (0 missing)
#>
#> Node number 30: 9 observations
#>   mean=2.581411, MSE=0.3669647
#>
#> Node number 31: 54 observations,    complexity param=0.01584638
#>   mean=4.461396, MSE=1.579623
#>   left son=62 (10 obs) right son=63 (44 obs)
#>   Primary splits:
#>       x2 < 1.130662  to the left,  improve=0.18386040, (0 missing)
#>       x1 < 6.209961  to the left,  improve=0.14561510, (0 missing)
#>       x3 < 4.517029  to the left,  improve=0.01044883, (0 missing)
#>
#> Node number 62: 10 observations
#>   mean=3.330957, MSE=2.001022
#>
#> Node number 63: 44 observations,    complexity param=0.01032524
#>   mean=4.718314, MSE=1.127413
#>   left son=126 (27 obs) right son=127 (17 obs)
#>   Primary splits:
#>       x1 < 6.468044  to the left,  improve=0.16079230, (0 missing)
#>       x3 < 5.608708  to the right, improve=0.05277854, (0 missing)
#>       x2 < 2.784688  to the left,  improve=0.03145241, (0 missing)
#>   Surrogate splits:
#>       x2 < 3.074905  to the left,  agree=0.636, adj=0.059, (0 split)
#>       x3 < 5.888028  to the left,  agree=0.636, adj=0.059, (0 split)
#>
#> Node number 126: 27 observations,    complexity param=0.01032524
#>   mean=4.380469, MSE=1.04313
#>   left son=252 (12 obs) right son=253 (15 obs)
#>   Primary splits:
#>       x1 < 3.658072  to the right, improve=0.4424566, (0 missing)
```

```
#>       x3 < 4.270123  to the right, improve=0.1430466, (0 missing)
#>       x2 < 2.658809  to the left,  improve=0.1121999, (0 missing)
#>   Surrogate splits:
#>       x2 < 2.707432  to the left,  agree=0.815, adj=0.583, (0 split)
#>       x3 < 4.010151  to the right, agree=0.593, adj=0.083, (0 split)
#>
#> Node number 127: 17 observations
#>   mean=5.25489, MSE=0.7920812
#>
#> Node number 252: 12 observations
#>   mean=3.620914, MSE=0.6204645
#>
#> Node number 253: 15 observations
#>   mean=4.988114, MSE=0.5504908

# Visualize the Regression Tree
rpart.plot(
    tree_model,
    type = 2,
    extra = 101,
    fallen.leaves = TRUE,
    main = "Regression Tree"
)
```

- Splits are made based on conditions (e.g., x1 $<$ 4.2), partitioning the predictor space.
- Terminal nodes (leaves) show the predicted value (mean response in that

region).
- The tree depth affects interpretability and overfitting risk.

```
# Optimal pruning based on cross-validation error
printcp(tree_model) # Displays CP table with cross-validation error
#>
#> Regression tree:
#> rpart(formula = y ~ x1 + x2 + x3, data = data_tree, method = "anova",
#>     control = rpart.control(cp = 0.01))
#>
#> Variables actually used in tree construction:
#> [1] x1 x2 x3
#>
#> Root node error: 989.7/100 = 9.897
#>
#> n= 100
#>
#>         CP nsplit rel error  xerror    xstd
#> 1 0.398959      0   1.00000 1.01348 0.34067
#> 2 0.174703      1   0.60104 0.86500 0.33363
#> 3 0.046074      2   0.42634 0.57079 0.18803
#> 4 0.027549      3   0.38026 0.52874 0.18667
#> 5 0.015846      4   0.35272 0.50611 0.18675
#> 6 0.010325      5   0.33687 0.51368 0.18610
#> 7 0.010000      7   0.31622 0.48471 0.18618
optimal_cp <-
    tree_model$cptable[which.min(tree_model$cptable[, "xerror"]), "CP"]

# Prune the tree
pruned_tree <- prune(tree_model, cp = optimal_cp)

# Visualize the pruned tree
rpart.plot(
    pruned_tree,
    type = 2,
    extra = 101,
    fallen.leaves = TRUE,
    main = "Pruned Regression Tree"
)
```

- Pruning reduces overfitting by simplifying the tree.
- The optimal CP minimizes cross-validation error, balancing complexity and fit.
- A shallower tree improves generalization on unseen data.

```
# Fit a Random Forest
set.seed(123)
rf_model <- randomForest(
    y ~ x1 + x2 + x3,
    data = data_tree,
    ntree = 500,
    mtry = 2,
    importance = TRUE
)

# Summary of the Random Forest
print(rf_model)
#>
#> Call:
#>  randomForest(formula = y ~ x1 + x2 + x3, data = data_tree, ntree = 500,      mtry = 2, impor
#>                Type of random forest: regression
#>                      Number of trees: 500
#> No. of variables tried at each split: 2
#>
#>           Mean of squared residuals: 3.031589
#>                     % Var explained: 69.37
```

- MSE decreases as more trees are added.

- % Variance Explained reflects predictive performance.
- `mtry = 2` indicates 2 random predictors are considered at each split.

```
# Plot OOB Error vs. Number of Trees
plot(rf_model, main = "Out-of-Bag Error for Random Forest")
```

- OOB error stabilizes as more trees are added, providing an unbiased estimate of test error.
- Helps determine if more trees improve performance or if the model has converged.

```
# Variable Importance
importance(rf_model)                # Numerical importance measures
#>      %IncMSE IncNodePurity
#> x1 10.145674     137.09918
#> x2  1.472662      77.41256
#> x3 44.232816     718.49567
varImpPlot(rf_model, main = "Variable Importance (Random Forest)")
```

- Mean Decrease in MSE indicates how much the model's error increases when a variable is permuted.
- Mean Decrease in Node Impurity reflects how much each variable reduces variance in splits.
- Variables with higher importance are more influential in the model.

```
# Predictions on new data
x_new <- seq(0, 10, length.out = 200)
test_data <- data.frame(x1 = x_new,
                        x2 = mean(x2),
                        x3 = mean(x3))

# Predictions
tree_pred <- predict(pruned_tree, newdata = test_data)
rf_pred <- predict(rf_model, newdata = test_data)

# Visualization
ggplot() +
    geom_point(aes(x1, y),
               data = data_tree,
               alpha = 0.5,
               color = "gray") +
    geom_line(
        aes(x_new, tree_pred),
        color = "blue",
        size = 1.2,
        linetype = "dashed"
```

```
    ) +
    geom_line(aes(x_new, rf_pred), color = "green", size = 1.2) +
    labs(
        title = "Regression Tree vs. Random Forest",
        subtitle = "Blue: Pruned Tree | Green: Random Forest",
        x = "x1",
        y = "Predicted y"
    ) +
    theme_minimal()
```

- The pruned regression tree (blue) shows step-like predictions, characteristic of piecewise constant fits.
- The random forest (green) provides a smoother fit by averaging across many trees, reducing variance.

```
# OOB Error (Random Forest)
oob_mse <- rf_model$mse[length(rf_model$mse)]  # Final OOB MSE

# Cross-Validation Error (Regression Tree)
cv_mse_tree <-
    min(tree_model$cptable[, "xerror"]) * var(data_tree$y)

# Compare OOB and CV errors
data.frame(
    Model = c("Pruned Regression Tree", "Random Forest"),
    MSE = c(cv_mse_tree, oob_mse)
)
```

```
#>                     Model      MSE
#> 1 Pruned Regression Tree 4.845622
#> 2          Random Forest 3.031589
```

- OOB error (Random Forest) provides an efficient, unbiased estimate without cross-validation.
- Cross-validation error (Regression Tree) evaluates generalization through resampling.
- Random Forest often shows lower MSE due to reduced variance.

10.9 Wavelet Regression

Wavelet regression is a nonparametric regression technique that represents the target function as a combination of **wavelet basis functions**. Unlike traditional basis functions (e.g., polynomials or splines), wavelets have excellent **localization properties** in both the **time (or space)** and **frequency** domains. This makes wavelet regression particularly effective for capturing **local features**, such as sharp changes, discontinuities, and transient patterns.

A wavelet is a function $\psi(t)$ that oscillates (like a wave) but is localized in both time and frequency. The key idea is to represent a function as a linear combination of shifted and scaled versions of a mother wavelet $\psi(t)$.

Wavelet basis functions are generated by scaling and translating the mother wavelet:

$$\psi_{j,k}(t) = 2^{j/2}\,\psi(2^j t - k),$$

where:

- j (scale parameter): Controls the **frequency**—larger j captures finer details (high-frequency components).
- k (translation parameter): Controls the **location**—shifting the wavelet along the time or space axis.
- The factor $2^{j/2}$ ensures that the wavelet basis functions are **orthonormal**.

In addition to the mother wavelet $\psi(t)$, there's also a **scaling function** $\phi(t)$, which captures the low-frequency (smooth) components of the data.

10.9.1 Wavelet Series Expansion

Just as Fourier analysis represents functions as sums of sines and cosines, **wavelet analysis** represents a function as a sum of wavelet basis functions:

$$f(t) = \sum_k c_{J_0,k}\, \phi_{J_0,k}(t) + \sum_{j=J_0}^{J_{\max}} \sum_k d_{j,k}\, \psi_{j,k}(t),$$

where:

- $c_{J_0,k}$ are the **approximation coefficients** at the coarsest scale J_0, capturing smooth trends,
- $d_{j,k}$ are the **detail coefficients** at scale j, capturing finer details,
- $\phi_{J_0,k}(t)$ are the scaling functions, and $\psi_{j,k}(t)$ are the wavelet functions.

The goal of wavelet regression is to estimate these coefficients based on observed data.

10.9.2 Wavelet Regression Model

Given data $\{(x_i, y_i)\}_{i=1}^n$, the wavelet regression model assumes:

$$y_i = f(x_i) + \varepsilon_i,$$

where:

- $f(x)$ is the unknown regression function,
- ε_i are i.i.d. errors with mean zero and variance σ^2.

We approximate $f(x)$ using a finite number of wavelet basis functions:

$$\hat{f}(x) = \sum_k \hat{c}_{J_0,k}\, \phi_{J_0,k}(x) + \sum_{j=J_0}^{J_{\max}} \sum_k \hat{d}_{j,k}\, \psi_{j,k}(x),$$

where the coefficients $\hat{c}_{J_0,k}$ and $\hat{d}_{j,k}$ are estimated from the data.

Coefficient Estimation:

- **Linear Wavelet Regression:** Estimate coefficients via least squares, projecting the data onto the wavelet basis.
- **Nonlinear Wavelet Regression (Thresholding):** Apply **shrinkage** or **thresholding** (e.g., hard or soft thresholding) to the detail coefficients $d_{j,k}$ to reduce noise and prevent overfitting. This is especially useful when the true signal is sparse in the wavelet domain.

10.9.3 Wavelet Shrinkage and Thresholding

Wavelet shrinkage is a powerful denoising technique introduced by Donoho and Johnstone (1995). The idea is to suppress small coefficients (likely to be noise) while retaining large coefficients (likely to contain the true signal).

Thresholding Rules:

1. **Hard Thresholding:**

$$\hat{d}_{j,k}^{(\text{hard})} = \begin{cases} d_{j,k}, & \text{if } |d_{j,k}| > \tau, \\ 0, & \text{otherwise,} \end{cases}$$

2. **Soft Thresholding:**

$$\hat{d}_{j,k}^{(\text{soft})} = \text{sign}(d_{j,k}) \cdot \max\left(|d_{j,k}| - \tau,\, 0\right),$$

where τ is the **threshold parameter**, often chosen based on the noise level (e.g., via cross-validation or universal thresholding).

Aspect	Kernel/Local Polynomial	Splines	Wavelets
Smoothness	Smooth, localized	Globally smooth with knots	Captures sharp discontinuities
Basis Functions	Kernel functions	Piecewise polynomials	Compact, oscillatory wavelets
Handling of Noise	Prone to overfitting without smoothing	Controlled via penalties	Excellent via thresholding
Computational Cost	Moderate	High (for large knots)	Efficient (fast wavelet transform)
Applications	Curve fitting, density estimation	Smoothing trends	Signal processing, time series

---

```r
# Simulated data: Noisy signal with discontinuities
set.seed(123)
n <- 96
x <- seq(0, 1, length.out = n)
signal <-
    sin(4 * pi * x) + ifelse(x > 0.5, 1, 0)  # Discontinuity at x = 0.5
y <- signal + rnorm(n, sd = 0.2)
```

```r
# Wavelet Regression using Discrete Wavelet Transform (DWT)
library(waveslim)

# Apply DWT with the correct parameter name
dwt_result <- dwt(y, wf = "haar", n.levels = 4)

# Thresholding (currently hard thresholding)
threshold <- 0.2
dwt_result_thresh <- dwt_result
for (i in 1:4) {
    dwt_result_thresh[[i]] <-
        ifelse(abs(dwt_result[[i]]) > threshold, dwt_result[[i]], 0)
}

# for soft thresholding
# for (i in 1:4) {
#     dwt_result_thresh[[i]] <-
#         sign(dwt_result[[i]]) * pmax(abs(dwt_result[[i]]) - threshold, 0)
# }


# Inverse DWT to reconstruct the signal
y_hat <- idwt(dwt_result_thresh)

# Plotting
plot(
    x,
    y,
    type = "l",
    col = "gray",
    lwd = 1,
    main = "Wavelet Regression (Denoising)"
)
lines(x,
      signal,
      col = "blue",
      lwd = 2,
      lty = 2)  # True signal
lines(x, y_hat, col = "red", lwd = 2)             # Denoised estimate
legend(
    "topright",
    legend = c("Noisy Data", "True Signal", "Wavelet Estimate"),
    col = c("gray", "blue", "red"),
    lty = c(1, 2, 1),
    lwd = c(1, 2, 2)
```

```
)
```

Advantages:

- **Local Feature Detection:** Excellent for capturing sharp changes, discontinuities, and localized phenomena.
- **Multiresolution Analysis:** Analyzes data at multiple scales, making it effective for both global trends and fine details.
- **Denoising Capability:** Powerful noise reduction through thresholding in the wavelet domain.

Limitations:

- **Complexity:** Requires careful selection of wavelet basis, decomposition levels, and thresholding methods.
- **Less Interpretability:** Coefficients are less interpretable compared to spline or tree-based methods.
- **Boundary Effects:** May suffer from artifacts near data boundaries without proper treatment.

```r
# Load necessary libraries
library(waveslim)   # For Discrete Wavelet Transform (DWT)
library(ggplot2)


# Simulate Data: Noisy signal with discontinuities
set.seed(123)
```

```r
n <- 96
x <- seq(0, 1, length.out = n)

# True signal: Sinusoidal with a discontinuity at x = 0.5
signal <- sin(4 * pi * x) + ifelse(x > 0.5, 1, 0)

# Add Gaussian noise
y <- signal + rnorm(n, sd = 0.2)

# Plot the noisy data and true signal
ggplot() +
    geom_line(aes(x, y),
              color = "gray",
              size = 0.8,
              alpha = 0.7) +
    geom_line(
        aes(x, signal),
        color = "blue",
        linetype = "dashed",
        size = 1
    ) +
    labs(title = "Noisy Data with Underlying True Signal",
         x = "x", y = "Signal") +
    theme_minimal()
```

```r
# Apply Discrete Wavelet Transform (DWT) using Haar wavelet
dwt_result <- dwt(y, wf = "haar", n.levels = 4)
```

```r
# View DWT structure
str(dwt_result)
#> List of 5
#>  $ d1: num [1:48] 0.14 -0.1221 0.3017 -0.1831 0.0745 ...
#>  $ d2: num [1:24] 0.50003 -0.06828 0.35298 0.01582 -0.00853 ...
#>  $ d3: num [1:12] 0.669 0.128 -0.746 -0.532 -0.258 ...
#>  $ d4: num [1:6] 1.07 -1.766 0.752 0.663 -1.963 ...
#>  $ s4: num [1:6] 2.963 -0.159 -2.758 6.945 3.845 ...
#>  - attr(*, "class")= chr "dwt"
#>  - attr(*, "wavelet")= chr "haar"
#>  - attr(*, "boundary")= chr "periodic"
```

- `wf = "haar"`: Haar wavelet, simple and effective for detecting discontinuities.
- `n.levels = 4`: Number of decomposition levels (captures details at different scales).

The DWT output contains approximation coefficients and detail coefficients for each level.

```r
# Hard Thresholding
threshold <- 0.2  # Chosen threshold for demonstration
dwt_hard_thresh <- dwt_result

# Apply hard thresholding to detail coefficients
for (i in 1:4) {
  dwt_hard_thresh[[i]] <- ifelse(abs(dwt_result[[i]]) > threshold, dwt_result[[i]], 0)
}
```

```r
# Soft Thresholding
dwt_soft_thresh <- dwt_result

# Apply soft thresholding to detail coefficients
for (i in 1:4) {
  dwt_soft_thresh[[i]] <- sign(dwt_result[[i]]) * pmax(abs(dwt_result[[i]]) - threshold, 0)
}
```

- Hard Thresholding: Keeps coefficients above the threshold, sets others to zero.
- Soft Thresholding: Shrinks coefficients toward zero, reducing potential noise smoothly.

```r
# Reconstruct the denoised signals using Inverse DWT
y_hat_hard <- idwt(dwt_hard_thresh)
y_hat_soft <- idwt(dwt_soft_thresh)
```

```r
# Combine data for ggplot
df_plot <- data.frame(
    x = rep(x, 4),
    y = c(y, signal, y_hat_hard, y_hat_soft),
    Type = rep(
        c(
            "Noisy Data",
            "True Signal",
            "Hard Thresholding",
            "Soft Thresholding"
        ),
        each = n
    )
)

# Plotting
ggplot(df_plot, aes(x, y, color = Type, linetype = Type)) +
    geom_line(size = 1) +
    scale_color_manual(values = c("gray", "blue", "red", "green")) +
    scale_linetype_manual(values = c("solid", "dashed", "solid", "solid")) +
    labs(
        title = "Wavelet Regression (Denoising)",
        subtitle = "Comparison of Hard vs. Soft Thresholding",
        x = "x",
        y = "Signal"
    ) +
    theme_minimal() +
    theme(legend.position = "top")
```

```
# Compute Mean Squared Error (MSE) for each method
mse_noisy <- mean((y - signal) ^ 2)
mse_hard <- mean((y_hat_hard - signal) ^ 2)
mse_soft <- mean((y_hat_soft - signal) ^ 2)

# Display MSE comparison
data.frame(
    Method = c("Noisy Data", "Hard Thresholding", "Soft Thresholding"),
    MSE = c(mse_noisy, mse_hard, mse_soft)
)
#>               Method        MSE
#> 1         Noisy Data 0.03127707
#> 2 Hard Thresholding 0.02814465
#> 3 Soft Thresholding 0.02267171
```

- Lower MSE indicates better denoising performance.
- Soft thresholding often achieves smoother results with lower MSE.

10.10 Multivariate Nonparametric Regression

Nonparametric regression in higher dimensions ($p > 1$) presents significant challenges compared to the univariate case. This difficulty arises primarily from the **curse of dimensionality**, which refers to the exponential growth of data requirements as the number of predictors increases.

10.10.1 The Curse of Dimensionality

The **curse of dimensionality** refers to various phenomena that occur when analyzing and organizing data in high-dimensional spaces. In the context of nonparametric regression:

- **Data sparsity:** As the number of dimensions increases, data become sparse. Even large datasets may not adequately cover the predictor space.
- **Exponential sample size growth:** To achieve the same level of accuracy, the required sample size grows exponentially with the number of dimensions. For example, to maintain the same density of points when moving from 1D to 2D, you need roughly the square of the sample size.

Illustration:

Consider estimating a function on the unit cube $[0,1]^p$. If we need 10 points per dimension to capture the structure:

- In 1D: 10 points suffice.
- In 2D: $10^2 = 100$ points are needed.
- In 10D: $10^{10} = 10,000,000,000$ points are required.

This makes traditional nonparametric methods, like kernel smoothing, impractical in high dimensions without additional structure or assumptions.

10.10.2 Multivariate Kernel Regression

A straightforward extension of kernel regression to higher dimensions is the multivariate Nadaraya-Watson estimator:

$$\widehat{m}_h(\mathbf{x}) = \frac{\sum_{i=1}^n K_h(\mathbf{x} - \mathbf{x}_i)\, y_i}{\sum_{i=1}^n K_h(\mathbf{x} - \mathbf{x}_i)},$$

where:

- $\mathbf{x} = (x_1, x_2, \dots, x_p)$ is the predictor vector,
- $K_h(\cdot)$ is a **multivariate kernel function**, often a product of univariate kernels:

$$K_h(\mathbf{x} - \mathbf{x}_i) = \prod_{j=1}^p K\left(\frac{x_j - x_{ij}}{h_j}\right),$$

- h_j is the bandwidth for the j-th predictor.

Challenges:

- The product kernel suffers from **inefficiency** in high dimensions because most data points are far from any given target point $\mathbf{x}$, resulting in very small kernel weights.
- Selecting an optimal **multivariate bandwidth matrix** is complex and computationally intensive.

10.10.3 Multivariate Splines

Extending splines to multiple dimensions involves more sophisticated techniques, as the simplicity of piecewise polynomials in 1D does not generalize easily.

10.10.3.1 Thin-Plate Splines

Thin-plate splines generalize cubic splines to higher dimensions. They minimize a smoothness penalty that depends on the derivatives of the function:

$$\hat{m}(\mathbf{x}) = \arg\min_f \left\{ \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2 + \lambda \int \|\nabla^2 f(\mathbf{x})\|^2 d\mathbf{x} \right\},$$

where $\nabla^2 f(\mathbf{x})$ is the **Hessian matrix** of second derivatives, and $\|\cdot\|^2$ represents the sum of squared elements.

Thin-plate splines are **rotation-invariant** and do not require explicit placement of knots, but they become computationally expensive as the number of dimensions increases.

10.10.3.2 Tensor Product Splines

For structured multivariate data, **tensor product splines** are commonly used. They construct a basis for each predictor and form the multivariate basis via tensor products:

$$\hat{m}(\mathbf{x}) = \sum_{i=1}^{K_1} \sum_{j=1}^{K_2} \beta_{ij} \, B_i(x_1) \, B_j(x_2),$$

where $B_i(x_1)$ and $B_j(x_2)$ are spline basis functions for x_1 and x_2, respectively.

Tensor products allow flexible modeling of **interactions** between predictors but can lead to large numbers of parameters as the number of dimensions increases.

10.10.4 Additive Models (GAMs)

A powerful approach to mitigating the curse of dimensionality is to assume an **additive structure** for the regression function:

$$m(\mathbf{x}) = \beta_0 + f_1(x_1) + f_2(x_2) + \cdots + f_p(x_p),$$

where each f_j is a univariate smooth function estimated nonparametrically.

10.10.4.1 Why Additive Models Help:

- **Dimensionality Reduction:** Instead of estimating a full p-dimensional surface, we estimate p separate functions in 1D.
- **Interpretability:** Each function $f_j(x_j)$ represents the effect of predictor X_j on the response, holding other variables constant.

Extensions:

- **Interactions:** Additive models can be extended to include interactions:

$$m(\mathbf{x}) = \beta_0 + \sum_{j=1}^{p} f_j(x_j) + \sum_{j<k} f_{jk}(x_j, x_k),$$

  where f_{jk} are bivariate smooth functions capturing interaction effects.

10.10.5 Radial Basis Functions

Radial basis functions (RBF) are another approach to multivariate nonparametric regression, particularly effective in scattered data interpolation.

A typical RBF model is:

$$\widehat{m}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \, \phi(\|\mathbf{x} - \mathbf{x}_i\|),$$

where:

- $\phi(\cdot)$ is a radial basis function (e.g., Gaussian: $\phi(r) = e^{-\gamma r^2}$),
- $\|\mathbf{x} - \mathbf{x}_i\|$ is the Euclidean distance between $\mathbf{x}$ and the data point $\mathbf{x}_i$,
- α_i are coefficients estimated from the data.

```r
# Load necessary libraries
library(ggplot2)
library(np)       # For multivariate kernel regression
```

```r
library(mgcv)     # For thin-plate and tensor product splines
library(fields)   # For radial basis functions (RBF)
library(reshape2) # For data manipulation


# Simulate Multivariate Data
set.seed(123)
n <- 100
x1 <- runif(n, 0, 5)
x2 <- runif(n, 0, 5)
x3 <- runif(n, 0, 5)

# True nonlinear multivariate function with interaction effects
true_function <- function(x1, x2, x3) {
    sin(pi * x1) * cos(pi * x2) + exp(-((x3 - 2.5) ^ 2)) + 0.5 * x1 * x3
}

# Response with noise
y <- true_function(x1, x2, x3) + rnorm(n, sd = 0.5)

# Data frame
data_multi <- data.frame(y, x1, x2, x3)

# Visualization of marginal relationships
p1 <-
    ggplot(data_multi, aes(x1, y)) +
    geom_point(alpha = 0.5) +
    labs(title = "Effect of x1")
p2 <-
    ggplot(data_multi, aes(x2, y)) +
    geom_point(alpha = 0.5) +
    labs(title = "Effect of x2")
p3 <-
    ggplot(data_multi, aes(x3, y)) +
    geom_point(alpha = 0.5) +
    labs(title = "Effect of x3")

gridExtra::grid.arrange(p1, p2, p3, ncol = 3)
```

```
# Multivariate Kernel Regression using np package
bw <-
    npregbw(y ~ x1 + x2 + x3, data = data_multi)  # Bandwidth selection
#>
Multistart 1 of 3 |
Multistart 1 of 3 |
Multistart 1 of 3 |
Multistart 1 of 3 /
Multistart 1 of 3 -
Multistart 1 of 3 |
Multistart 1 of 3 |
Multistart 1 of 3 /
Multistart 1 of 3 -
Multistart 1 of 3 \
Multistart 2 of 3 |
Multistart 2 of 3 |
Multistart 2 of 3 /
Multistart 2 of 3 -
Multistart 2 of 3 \
Multistart 2 of 3 |
Multistart 2 of 3 |
Multistart 2 of 3 |
Multistart 2 of 3 /
Multistart 3 of 3 |
Multistart 3 of 3 |
Multistart 3 of 3 /
Multistart 3 of 3 -
```

```
Multistart 3 of 3 \
Multistart 3 of 3 |
Multistart 3 of 3 |
Multistart 3 of 3 |
Multistart 3 of 3 /

kernel_model <- npreg(bws = bw)

# Predict on a grid for visualization
grid_data <- expand.grid(
    x1 = seq(0, 5, length.out = 50),
    x2 = seq(0, 5, length.out = 50),
    x3 = mean(data_multi$x3)
)
pred_kernel <- predict(kernel_model, newdata = grid_data)

# Visualization
grid_data$pred <- pred_kernel
ggplot(grid_data, aes(x1, x2, fill = pred)) +
    geom_raster() +
    scale_fill_viridis_c() +
    labs(title = "Multivariate Kernel Regression (x3 fixed)",
         x = "x1",
         y = "x2") +
    theme_minimal()
```

- Kernel regression captures nonlinear interactions but suffers from data

sparsity in high dimensions (curse of dimensionality).
- Bandwidth selection is critical for performance.

```
# Fit Thin-Plate Spline
tps_model <- gam(y ~ s(x1, x2, x3, bs = "tp", k = 5), data = data_multi)

# Predictions
grid_data$pred_tps <- predict(tps_model, newdata = grid_data)

# Visualization
ggplot(grid_data, aes(x1, x2, fill = pred_tps)) +
    geom_raster() +
    scale_fill_viridis_c() +
    labs(title = "Thin-Plate Spline (x3 fixed)",
         x = "x1", y = "x2") +
    theme_minimal()
```

- Thin-plate splines handle smooth surfaces well and are rotation-invariant.
- Computational cost increases with higher dimensions due to matrix operations.

```
# Fit Tensor Product Spline
tensor_model <- gam(y ~ te(x1, x2, x3), data = data_multi)

# Predictions
grid_data$pred_tensor <- predict(tensor_model, newdata = grid_data)

# Visualization
```

```r
ggplot(grid_data, aes(x1, x2, fill = pred_tensor)) +
    geom_raster() +
    scale_fill_viridis_c() +
    labs(title = "Tensor Product Spline (x3 fixed)",
         x = "x1", y = "x2") +
    theme_minimal()
```

- Tensor product splines model interactions explicitly between variables.
- Suitable when data have structured dependencies but can lead to many parameters in higher dimensions.

```r
# Additive Model (GAM)
gam_model <- gam(y ~ s(x1) + s(x2) + s(x3), data = data_multi)

# Predictions
grid_data$pred_gam <- predict(gam_model, newdata = grid_data)

# Visualization
ggplot(grid_data, aes(x1, x2, fill = pred_gam)) +
    geom_raster() +
    scale_fill_viridis_c() +
    labs(title = "Additive Model (GAM, x3 fixed)",
         x = "x1", y = "x2") +
    theme_minimal()
```

Why GAMs Help:

- Dimensionality reduction: Instead of estimating a full multivariate function, GAM estimates separate 1D functions for each predictor.
- Interpretability: Easy to understand individual effects.
- Limitations: Cannot capture complex interactions unless explicitly added.

```
# Radial Basis Function Model
rbf_model <- Tps(cbind(x1, x2, x3), y)  # Thin-plate spline RBF
#> Warning:
#> Grid searches over lambda (nugget and sill variances) with  minima at the endpoints
#>   (GCV) Generalized Cross-Validation
#>    minimum at  right endpoint  lambda  =  0.0002622876 (eff. df= 95.00001 )

# Predictions
grid_data$pred_rbf <-
    predict(rbf_model, cbind(grid_data$x1, grid_data$x2, grid_data$x3))

# Visualization
ggplot(grid_data, aes(x1, x2, fill = pred_rbf)) +
    geom_raster() +
    scale_fill_viridis_c() +
    labs(title = "Radial Basis Function Regression (x3 fixed)",
         x = "x1", y = "x2") +
    theme_minimal()
```

- RBFs capture complex, smooth surfaces and interactions based on distance.
- Perform well for scattered data but can be computationally expensive for large datasets.

```
# Compute Mean Squared Error for each model
mse_kernel <- mean((predict(kernel_model) - data_multi$y) ^ 2)
mse_tps <- mean((predict(tps_model) - data_multi$y) ^ 2)
mse_tensor <- mean((predict(tensor_model) - data_multi$y) ^ 2)
mse_gam <- mean((predict(gam_model) - data_multi$y) ^ 2)
mse_rbf <-
    mean((predict(rbf_model, cbind(x1, x2, x3)) - data_multi$y) ^ 2)

# Display MSE comparison
data.frame(
    Model = c(
        "Kernel Regression",
        "Thin-Plate Spline",
        "Tensor Product Spline",
        "Additive Model (GAM)",
        "Radial Basis Functions"
    ),
    MSE = c(mse_kernel, mse_tps, mse_tensor, mse_gam, mse_rbf)
)
#>                     Model         MSE
#> 1       Kernel Regression 0.253019836
#> 2       Thin-Plate Spline 0.449096723
```

```
#> 3  Tensor Product Spline 0.304680406
#> 4   Additive Model (GAM) 1.595616092
#> 5 Radial Basis Functions 0.001361915
```

10.11 Conclusion: The Evolving Landscape of Regression Analysis

As we conclude this exploration of regression analysis, we reflect on the vast landscape we have navigated—spanning from the foundational principles of **linear regression** to the intricate complexities of **generalized linear models**, **linear mixed models**, **nonlinear mixed models**, and now, the flexible world of **nonparametric regression**.

Regression analysis is more than just a statistical tool; it is a **versatile framework** that underpins decision-making across disciplines—from marketing and finance to healthcare, engineering, and beyond. This journey has shown how regression serves not only as a method for modeling relationships but also as a lens through which we interpret complex data in an ever-changing world.

10.11.1 Key Takeaways

- **The Power of Simplicity:** At its core, simple **linear regression** illustrates how relationships between variables can be modeled with clarity and elegance. Mastering these fundamentals lays the groundwork for more complex techniques.
- **Beyond Linearity: Nonlinear regression** and **generalized linear models** extend our capabilities to handle data that defy linear assumptions—capturing curved relationships, non-normal error structures, and diverse outcome distributions.
- **Accounting for Hierarchies and Dependencies:** Real-world data often exhibit structures such as nested observations or repeated measures. **Linear mixed models** and **generalized linear mixed models** enable us to account for both fixed effects and random variability, ensuring robust and nuanced inferences.
- **Complex Systems, Flexible Models: Nonlinear mixed models** allow us to capture dynamic, non-linear processes with hierarchical structures, bridging the gap between theoretical models and real-world complexity.
- **The Flexibility of Nonparametric Regression: Nonparametric methods**, such as **kernel regression**, **local polynomial regression**,

smoothing splines, **wavelet regression**, and **regression trees**, provide powerful tools when parametric assumptions are too restrictive. These models excel at capturing complex, nonlinear patterns without assuming a specific functional form, offering greater adaptability in diverse applications.

10.11.2 The Art and Science of Regression

While statistical formulas and algorithms form the backbone of regression analysis, the **true art** lies in **model selection**, **diagnostic evaluation**, and **interpretation**. No model is inherently perfect; each is an approximation of reality, shaped by the assumptions we make and the data we collect. The most effective analysts are those who approach models critically—testing assumptions, validating results, and recognizing the limitations of their analyses.

Nonparametric methods remind us that **flexibility** often comes at the cost of **interpretability** and **efficiency**, just as parametric models offer simplicity but may risk oversimplification. The key is not to choose between these paradigms, but to understand when each is most appropriate.

10.11.3 Looking Forward

The field of regression continues to evolve, driven by rapid advancements in **computational power**, **data availability**, and **methodological innovation**. This evolution has given rise to a wide range of modern techniques that extend beyond traditional frameworks:

- **Machine Learning Algorithms:** While methods like random forests, support vector machines, and gradient boosting are well-established, recent developments include:
  - **Extreme Gradient Boosting (XGBoost)** and **LightGBM**, optimized for speed and performance in large-scale data environments.
  - **CatBoost**, which handles categorical features more effectively without extensive preprocessing.
- **Bayesian Regression Techniques:** Modern Bayesian approaches go beyond simple hierarchical models to include:
  - **Bayesian Additive Regression Trees (BART):** A flexible, nonparametric Bayesian method that combines the power of regression trees with probabilistic inference.
  - **Bayesian Neural Networks (BNNs):** Extending deep learning with uncertainty quantification, enabling robust decision-making in high-stakes applications.
- **High-Dimensional Data Analysis:** Regularization methods like LASSO and ridge regression have paved the way for more advanced

techniques, such as:
  - **Graphical Models and Sparse Precision Matrices:** For capturing complex dependency structures in high-dimensional data.
- **Deep Learning for Regression:** Deep neural networks (DNNs) are increasingly used for regression tasks, particularly when dealing with:
  - **Structured Data (e.g., tabular datasets):** Through architectures like **TabNet**.
  - **Unstructured Data (e.g., images, text):** Using **convolutional neural networks (CNNs)** and **transformer-based models**.
- **Causal Inference in Regression:** The integration of causal modeling techniques into regression frameworks has advanced significantly:
  - **Double Machine Learning (DML):** Combining machine learning with econometric methods for robust causal effect estimation.
  - **Causal Forests:** An extension of random forests designed to estimate heterogeneous treatment effects.
- **Functional Data Analysis (FDA):** For analyzing data where predictors or responses are functions (e.g., curves, time series), using methods like:
  - **Functional Linear Models (FLM)** and **Functional Additive Models (FAM)**.
  - **Dynamic Regression Models** for real-time prediction in streaming data environments.

While these modern approaches differ in implementation, many are rooted in the fundamental concepts covered in this book. Whether through parametric precision or nonparametric flexibility, the principles of regression remain central to data-driven inquiry.

10.11.4 Final Thoughts

As you apply these techniques in your own work, remember that regression is not just about fitting models—it's about:

- **Asking the right questions**
- **Interpreting results thoughtfully**
- **Using data to generate meaningful insights**

Whether you're developing marketing strategies, forecasting financial trends, optimizing healthcare interventions, or conducting academic research, the tools you've gained here will serve as a strong foundation.

> In the words of **George E.P. Box**,
> *"All models are wrong, but some are useful."*
>
> Our goal as analysts is to find models that are not only useful but also **enlightening**—models that reveal patterns, guide decisions, and deepen our understanding of the world.

III. RAMIFICATIONS

Chapter 11

Data

Data can be defined broadly as any set of values, facts, or statistics that can be used for reference, analysis, and drawing inferences. In research, data drives the process of understanding phenomena, testing hypotheses, and formulating evidence-based conclusions. Choosing the right type of data (and understanding its strengths and limitations) is critical for the validity and reliability of findings.

11.1 Data Types

11.1.1 Qualitative vs. Quantitative Data

A foundational way to categorize data is by whether it is **qualitative** (non-numerical) or **quantitative** (numerical). These distinctions often guide research designs, data collection methods, and analytical techniques.

Qualitative	**Quantitative**
Examples: In-depth interviews, focus groups, case studies, ethnographies, open-ended questions, field notes	**Examples**: Surveys with closed-ended questions, experiments, numerical observations, structured interviews
Nature: Text-based, often descriptive, subjective interpretations	**Nature**: Numeric, more standardized, objective measures
Analysis: Thematic coding, content analysis, discourse analysis	**Analysis**: Statistical tests, regression, hypothesis testing, descriptive statistics
Outcome: Rich context, detailed understanding of phenomena	**Outcome**: Measurable facts, generalizable findings (with appropriate sampling and design)

11.1.1.1 Uses and Advantages of Qualitative Data

- **Deep Understanding**: Captures context, motivations, and perceptions in depth.
- **Flexibility**: Elicits new insights through open-ended inquiry.
- **Inductive Approaches**: Often used to build new theories or conceptual frameworks.

11.1.1.2 Uses and Advantages of Quantitative Data

- **Measurement and Comparison**: Facilitates measuring variables and comparing across groups or over time.
- **Generalizability**: With proper sampling, findings can often be generalized to broader populations.
- **Hypothesis Testing**: Permits the use of statistical methods to test specific predictions or relationships.

11.1.1.3 Limitations of Qualitative and Quantitative Data

- **Qualitative**:
  - Findings may be difficult to generalize if samples are small or non-representative.
  - Analysis can be time-consuming due to coding and interpreting text.
  - Potential for researcher bias in interpretation.
- **Quantitative**:
  - May oversimplify complex human behaviors or contextual factors by reducing them to numbers.
  - Validity depends heavily on how well constructs are operationalized.
  - Can miss underlying meanings or nuances not captured in numeric measures.

11.1.1.4 Levels of Measurement

Even within **quantitative** data, there are further distinctions based on the *level of measurement*. This classification is crucial for determining which statistical techniques are appropriate:

1. **Nominal**: Categorical data with no inherent order (e.g., gender, blood type, eye color).
2. **Ordinal**: Categorical data with a specific order or ranking but without consistent intervals between ranks (e.g., Likert scale responses: "strongly disagree," "disagree," "neutral," "agree," "strongly agree").

3. **Interval**: Numeric data with equal intervals but no true zero (e.g., temperature in Celsius or Fahrenheit).
4. **Ratio**: Numeric data with equal intervals and a meaningful zero (e.g., height, weight, income).

The level of measurement affects which statistical tests (like t-tests, ANOVA, correlations, regressions) are valid and how you can interpret differences or ratios in the data.

11.1.2 Other Ways to Classify Data

Beyond **observational structure**, there are multiple other dimensions used to classify data:

11.1.2.1 Primary vs. Secondary Data

- **Primary Data**: Collected directly by the researcher for a specific purpose (e.g., firsthand surveys, experiments, direct measurements).
- **Secondary Data**: Originally gathered by someone else for a different purpose (e.g., government census data, administrative records, previously published datasets).

11.1.2.2 Structured, Semi-Structured, and Unstructured Data

- **Structured Data**: Organized in a predefined manner, typically in rows and columns (e.g., spreadsheets, relational databases).
- **Semi-Structured Data**: Contains organizational markers but not strictly tabular (e.g., JSON, XML logs, HTML).
- **Unstructured Data**: Lacks a clear, consistent format (e.g., raw text, images, videos, audio files).
  - Often analyzed using natural language processing (NLP), image recognition, or other advanced techniques.

11.1.2.3 Big Data

- Characterized by the "3 Vs": **Volume** (large amounts), **Variety** (diverse forms), and **Velocity** (high-speed generation).
- Requires specialized computational tools (e.g., Hadoop, Spark) and often cloud-based infrastructure for storage and processing.
- Can be structured or unstructured (e.g., social media feeds, sensor data, clickstream data).

11.1.2.4 Internal vs. External Data (in Organizational Contexts)

- **Internal Data**: Generated within an organization (e.g., sales records, HR data, production metrics).
- **External Data**: Sourced from outside (e.g., macroeconomic indicators, market research reports, social media analytics).

11.1.2.5 Proprietary vs. Public Datas

- **Proprietary Data**: Owned by an organization or entity, not freely available for public use.
- **Public/Open Data**: Freely accessible data provided by governments, NGOs, or other institutions (e.g., data.gov, World Bank Open Data).

11.1.3 Data by Observational Structure Over Time

Another primary way to categorize data is by *how* observations are collected over time. This classification shapes research design, analytic methods, and the types of inferences we can make. Four major types here are:

1. Cross-Sectional Data
2. Time Series Data
3. Repeated Cross-Sectional Data
4. Panel (Longitudinal) Data

Type	Advantages	Limitations
Cross-Sectional Data	Simple, cost-effective, good for studying distributions or correlations at a single time point.	Lacks temporal information, can only infer associations, not causal links.
Time Series Data	Enables trend analysis, seasonality detection, and forecasting.	Requires handling autocorrelation, stationarity issues, and structural breaks.
Repeated Cross-Sectional Data	Tracks shifts in population-level parameters over time; simpler than panel data.	Cannot track individual changes; comparability depends on consistent methodology.
Panel (Longitudinal) Data	Allows causal inference, controls for unobserved heterogeneity, tracks individual trajectories.	Expensive, prone to attrition, requires complex statistical methods.

11.2 Cross-Sectional Data

Cross-sectional data consists of observations on **multiple entities** (e.g., individuals, firms, regions, or countries) at **a single point in time** or over a very short period, where **time is not a primary dimension** of variation.

- Each observation represents a **different entity**, rather than the same entity tracked over time.
- Unlike time series data, the order of observations does not carry temporal meaning.

Examples

- **Labor Economics**: Wage and demographic data for 1,000 workers in 2024.
- **Marketing Analytics**: Customer satisfaction ratings and purchasing behavior for 500 online shoppers surveyed in Q1 of a year.
- **Corporate Finance**: Financial statements of 1,000 firms for the fiscal year 2023.

Key Characteristics

- **Observations are independent (in an ideal setting)**: Each unit is drawn from a population with no intrinsic dependence on others.
- **No natural ordering**: Unlike time series data, the sequence of observations does not affect analysis.
- **Variation occurs across entities, not over time**: Differences in observed outcomes arise from differences between individuals, firms, or regions.

Advantages

- **Straightforward Interpretation**: Since time effects are not present, the focus remains on relationships between variables at a single point.
- **Easier to Collect and Analyze**: Compared to time series or panel data, cross-sectional data is often simpler to collect and model.
- **Suitable for causal inference** (if exogeneity conditions hold).

Challenges

- **Omitted Variable Bias**: Unobserved confounders may drive both the dependent and independent variables.
- **Endogeneity**: Reverse causality or measurement error can introduce bias.
- **Heteroskedasticity**: Variance of errors may differ across entities, requiring robust standard errors.

A typical cross-sectional regression model:

$$y_i = \beta_0 + x_{i1}\beta_1 + x_{i2}\beta_2 + \cdots + x_{i(k-1)}\beta_{k-1} + \epsilon_i$$

where:

- y_i is the outcome variable for entity i,
- x_{ij} are explanatory variables,
- ϵ_i is an error term capturing unobserved factors.

11.3 Time Series Data

Time series data consists of observations on the *same variable(s)* recorded over multiple time periods for a single entity (or aggregated entity). These data points are typically collected at consistent intervals—hourly, daily, monthly, quarterly, or annually—allowing for the analysis of trends, patterns, and forecasting.

Examples

- **Stock Market**: Daily closing prices of a company's stock over five years.
- **Economics**: Monthly unemployment rates in a country over a decade.
- **Macroeconomics**: Annual GDP of a country from 1960 to 2020.

Key Characteristics

- The primary goal is to analyze trends, seasonality, cyclic patterns, and forecast future values.
- Time series data requires specialized statistical methods, such as:
  - **Autoregressive Integrated Moving Average (ARIMA)**
  - **Seasonal ARIMA (SARIMA)**
  - **Exponential Smoothing**
  - **Vector Autoregression (VAR)**

Advantages

- Captures temporal patterns such as trends, seasonal fluctuations, and economic cycles.
- Essential for forecasting and policy-making, such as setting interest rates based on economic indicators.

Challenges

- **Autocorrelation**: Observations close in time are often correlated.
- **Structural Breaks**: Sudden changes due to policy shifts or economic crises can distort analysis.
- **Seasonality**: Must be accounted for to avoid misleading conclusions.

A time series typically consists of four key components:

1. **Trend**: Long-term directional movement in the data over time.
2. **Seasonality**: Regular, periodic fluctuations (e.g., increased retail sales in December).
3. **Cyclical Patterns**: Long-term economic cycles that are irregular but recurrent.
4. **Irregular (Random) Component**: Unpredictable variations not explained by trend, seasonality, or cycles.

A general linear time series model can be expressed as:

$$y_t = \beta_0 + x_{t1}\beta_1 + x_{t2}\beta_2 + \cdots + x_{t(k-1)}\beta_{k-1} + \epsilon_t$$

Some Common Model Types

1. **Static Model**

A simple time series regression:

$$y_t = \beta_0 + x_1\beta_1 + x_2\beta_2 + x_3\beta_3 + \epsilon_t$$

2. **Finite Distributed Lag Model**

Captures the effect of past values of an explanatory variable:

$$y_t = \beta_0 + pe_t\delta_0 + pe_{t-1}\delta_1 + pe_{t-2}\delta_2 + \epsilon_t$$

- **Long-Run Propensity**: Measures the cumulative effect of explanatory variables over time:

$$LRP = \delta_0 + \delta_1 + \delta_2$$

3. **Dynamic Model**

A model incorporating lagged dependent variables:

$$GDP_t = \beta_0 + \beta_1 GDP_{t-1} + \epsilon_t$$

11.3.1 Statistical Properties of Time Series Models

For time series regression, standard OLS assumptions must be carefully examined. The following conditions affect estimation:

Finite Sample Properties

- **A1-A3**: OLS remains unbiased.
- **A1-A4**: Standard errors are consistent, and the Gauss-Markov Theorem holds (OLS is BLUE).
- **A1-A6**: Finite sample Wald tests (e.g., t-tests and F-tests) remain valid.

However, in time series settings, A3 often fails due to:

- **Spurious Time Trends** (fixable by including a time trend)
- **Strict vs. Contemporaneous Exogeneity** (sometimes unavoidable)

11.3.2 Common Time Series Processes

Several key models describe different time series behaviors:

- **Autoregressive Model (AR(p))**: A process where current values depend on past values.
- **Moving Average Model (MA(q))**: A process where past error terms influence current values.
- **Autoregressive Moving Average (ARMA(p, q))**: A combination of AR and MA processes.
- **Autoregressive Conditional Heteroskedasticity (ARCH(p))**: Models time-varying volatility.
- **Generalized ARCH (GARCH(p, q))**: Extends ARCH by including past conditional variances.

11.3.3 Deterministic Time Trends

When both the dependent and independent variables exhibit trending behavior, a regression may produce **spurious** results.

Spurious Regression Example

A simple regression with trending variables:

$$y_t = \alpha_0 + t\alpha_1 + v_t$$

$$x_t = \lambda_0 + t\lambda_1 + u_t$$

where

- $\alpha_1 \neq 0$ and $\lambda_1 \neq 1$
- v_t and u_t are independent.

Despite no true relationship between x_t and y_t, estimating:

$$y_t = \beta_0 + x_t\beta_1 + \epsilon_t$$

results in:

- **Inconsistency**: $plim(\hat{\beta}_1) = \frac{\alpha_1}{\lambda_1}$
- **Invalid Inference**: $|t| \to^d \infty$ for $H_0 : \beta_1 = 0$, leading to rejection of the null hypothesis as $n \to \infty$.
- **Misleading** R^2: $plim(R^2) = 1$, falsely implying perfect predictive power.

We can also rewrite the equation as:

$$\begin{aligned} y_t &= \beta_0 + \beta_1 x_t + \epsilon_t \\ \epsilon_t &= \alpha_1 t + v_t \end{aligned}$$

where $\beta_0 = \alpha_0$ and $\beta_1 = 0$. Since x_t is a deterministic function of time, ϵ_t is correlated with x_t, leading to the usual omitted variable bias.

Solutions to Spurious Trends

1. **Include a Time Trend** (t) as a Control Variable
   - Provides consistent parameter estimates and valid inference.
2. **Detrend Variables**
   - Regress both y_t and x_t on time, then use residuals in a second regression.
   - Equivalent to applying the Frisch-Waugh-Lovell Theorem.

11.3.4 Violations of Exogeneity in Time Series Models

The **exogeneity assumption** (A3) plays a crucial role in ensuring unbiased and consistent estimation in time series models. However, in many cases, the assumption is **violated** due to the inherent nature of time-dependent processes.

In a standard regression framework, we assume:

$$E(\epsilon_t | x_1, x_2, ..., x_T) = 0$$

which requires that the error term is uncorrelated with all past, present, and future values of the independent variables.

Common Violations of Exogeneity

1. Feedback Effect

- The error term ϵ_t **influences future values** of the independent variables.
- A classic example occurs in economic models where past shocks affect future decisions.

2. Dynamic Specification

- The dependent variable includes a **lagged version of itself** as an explanatory variable, introducing correlation between ϵ_t and past y_{t-1}.

3. Dynamic Completeness

- In finite distributed lag (FDL) models, failing to include the **correct number of lags** leads to omitted variable bias and correlation between regressors and errors.

11.3.4.1 Feedback Effect

In a simple regression model:

$$y_t = \beta_0 + x_t\beta_1 + \epsilon_t$$

the standard exogeneity assumption (A3) requires:

$$E(\epsilon_t|x_1, x_2, ..., x_t, x_{t+1}, ..., x_T) = 0$$

However, in the presence of feedback, past errors affect future values of x_t, leading to:

$$E(\epsilon_t|x_{t+1}, ..., x_T) \neq 0$$

- This occurs when **current shocks** (e.g., economic downturns) influence **future decisions** (e.g., government spending, firm investments).
- Strict exogeneity is **violated**, as we now have dependence across time.

Implication:

- Standard OLS estimators become **biased and inconsistent**.
- One common solution is using Instrumental Variables to isolate exogenous variation in x_t.

11.3.4.2 Dynamic Specification

A **dynamically specified model** includes lagged dependent variables:

$$y_t = \beta_0 + y_{t-1}\beta_1 + \epsilon_t$$

Exogeneity (A3) would require:

$$E(\epsilon_t | y_1, y_2, ..., y_t, y_{t+1}, ..., y_T) = 0$$

However, since y_{t-1} depends on ϵ_{t-1} from the previous period, we obtain:

$$Cov(y_{t-1}, \epsilon_t) \neq 0$$

Implication:

- Strict exogeneity (A3) fails, as y_{t-1} and ϵ_t are correlated.
- OLS estimates are biased and inconsistent.
- Standard **autoregressive models (AR)** require alternative estimation techniques like **Generalized Method of Moments** or Maximum Likelihood Estimation.

11.3.4.3 Dynamic Completeness and Omitted Lags

A finite distributed lag (FDL) model:

$$y_t = \beta_0 + x_t \delta_0 + x_{t-1} \delta_1 + \epsilon_t$$

assumes that the included lags **fully capture** the relationship between y_t and past values of x_t. However, if we omit relevant lags, the exogeneity assumption (A3):

$$E(\epsilon_t | x_1, x_2, ..., x_t, x_{t+1}, ..., x_T) = 0$$

fails, as unmodeled lag effects create correlation between x_{t-2} and ϵ_t.

Implication:

- The regression suffers from **omitted variable bias**, making OLS estimates unreliable.
- **Solution**:
  - Include **additional lags** of x_t.
  - Use **lag selection criteria** (e.g., AIC, BIC) to determine the appropriate lag structure.

11.3.5 Consequences of Exogeneity Violations

If strict exogeneity (A3) fails, standard OLS assumptions no longer hold:

- OLS is biased.
- Gauss-Markov Theorem no longer applies.

- Finite Sample Properties (such as unbiasedness) are invalid.

To address these issues, we can:

1. Rely on Large Sample Properties: Under certain conditions, consistency may still hold.
2. Use Weaker Forms of Exogeneity: Shift from **strict exogeneity** (A3) to **contemporaneous exogeneity** (A3a).

If strict exogeneity does not hold, we can instead assume A3a (Contemporaneous Exogeneity):

$$E(\mathbf{x}_t'\epsilon_t) = 0$$

This weaker assumption only requires that x_t is uncorrelated with the error in the same time period.

Key Differences from Strict Exogeneity

Exogeneity Type	Requirement	Allows Dynamic Models?
Strict Exogeneity	$E(\epsilon_t \mid x_1, x_2, ..., x_T) = 0$	**No**
Contemporaneous Exogeneity	$E(\mathbf{x}_t'\epsilon_t) = 0$	**Yes**

- With contemporaneous exogeneity, ϵ_t can be correlated with past and future values of x_t.
- This allows for dynamic specifications such as:

$$y_t = \beta_0 + y_{t-1}\beta_1 + \epsilon_t$$

  while still maintaining **consistency** under certain assumptions.

Deriving Large Sample Properties for Time Series

To establish **consistency** and **asymptotic normality**, we rely on the following assumptions:

- A1: Linearity
- A2: Full Rank (No Perfect Multicollinearity)
- A3a: Contemporaneous Exogeneity

However, the standard Weak Law of Large Numbers and Central Limit Theorem in OLS depend on A5 (Random Sampling), which does **not** hold in time series settings.

Since time series data exhibits dependence over time, we replace A5 (Random Sampling) with a weaker assumption:

- A5a: Weak Dependence (Stationarity)

Asymptotic Variance and Serial Correlation

- The derivation of **asymptotic variance** depends on A4 (Homoskedasticity).
- However, in time series settings, we often encounter **serial correlation**:

$$Cov(\epsilon_t, \epsilon_s) \neq 0 \quad \text{for} \quad |t - s| > 0$$

- To ensure valid inference, standard errors must be corrected using methods such as Newey-West HAC estimators.

11.3.6 Highly Persistent Data

In time series analysis, a key assumption for OLS consistency is that the data-generating process exhibits A5a weak dependence (i.e., observations are not too strongly correlated over time). However, when y_t and x_t are highly persistent, standard OLS assumptions break down.

If a time series is **not weakly dependent**, it means:

- y_t and y_{t-h} remain strongly correlated even for large lags ($h \to \infty$).
- A5a (Weak Dependence Assumption) fails, leading to:
  - **OLS inconsistency**.
  - **No valid limiting distribution** (asymptotic normality does not hold).

Example: A classic example of a highly persistent process is a random walk:

$$y_t = y_{t-1} + u_t$$

or with drift:

$$y_t = \alpha + y_{t-1} + u_t$$

where u_t is a white noise error term.

- y_t does not revert to a mean—it has an infinite variance as $t \to \infty$.

- **Shocks accumulate**, making standard regression analysis unreliable.

11.3.6.1 Solution: First Differencing

A common way to transform non-stationary series into stationary ones is through first differencing:

$$\Delta y_t = y_t - y_{t-1} = u_t$$

- If u_t is a weakly dependent process (i.e., $I(0)$, stationary), then y_t is said to be **difference-stationary** or **integrated of order 1,** $I(1)$.
- If both y_t and x_t follow a random walk ($I(1)$), we estimate:

$$\begin{aligned}\Delta y_t &= (\Delta \mathbf{x}_t \beta) + (\epsilon_t - \epsilon_{t-1}) \\ \Delta y_t &= \Delta \mathbf{x}_t \beta + \Delta u_t\end{aligned}$$

This ensures **OLS estimation remains valid**.

11.3.7 Unit Root Testing

To formally determine whether a time series contains a **unit root** (i.e., is non-stationary), we test:

$$y_t = \alpha + \rho y_{t-1} + u_t$$

Hypothesis Testing

- $H_0 : \rho = 1$ (unit root, non-stationary)
  - OLS is **not consistent or asymptotically normal**.
- $H_a : \rho < 1$ (stationary process)
  - OLS is **consistent and asymptotically normal**.

Key Issues

- The usual **t-test is not valid** because OLS under H_0 does not have a standard distribution.
- Instead, specialized tests such as Dickey-Fuller and Augmented Dickey-Fuller tests are required.

11.3.7.1 Dickey-Fuller Test for Unit Roots

The **Dickey-Fuller test** transforms the original equation by subtracting y_{t-1} from both sides:

$$\Delta y_t = \alpha + \theta y_{t-1} + v_t$$

where:

$$\theta = \rho - 1$$

- **Null Hypothesis** ($H_0 : \theta = 0$) $\rightarrow$ Implies $\rho = 1$ (unit root, non-stationary).
- **Alternative** ($H_a : \theta < 0$) $\rightarrow$ Implies $\rho < 1$ (stationary).

Since y_t follows a non-standard asymptotic distribution under H_0, Dickey and Fuller derived specialized critical values.

Decision Rule

- If the test statistic is more negative than the critical value, reject $H_0 \rightarrow y_t$ is stationary.
- Otherwise, fail to reject $H_0 \rightarrow y_t$ has a unit root (non-stationary).

The standard DF test may fail due to two key limitations:

1. **Simplistic Dynamic Relationship**

- The DF test assumes only one lag in the autoregressive structure.
- However, in reality, higher-order lags of Δy_t may be needed.

Solution:
Use the Augmented Dickey-Fuller test, which includes extra lags:

$$\Delta y_t = \alpha + \theta y_{t-1} + \gamma_1 \Delta y_{t-1} + \cdots + \gamma_p \Delta y_{t-p} + v_t$$

- Under H_0, Δy_t follows an AR(1) process.
- Under H_a, y_t follows an AR(2) or higher process.

Including lags of Δy_t ensures a better-specified model.

2. **Ignoring Deterministic Time Trends**

If a series exhibits a deterministic trend, failing to include it biases the unit root test.

Example: If y_t grows over time, a test without a trend component will falsely detect a unit root.

Solution: Include a deterministic time trend (t) in the regression:

$$\Delta y_t = \alpha + \theta y_{t-1} + \delta t + v_t$$

- Allows for quadratic relationships with time.
- Changes the critical values, requiring an adjusted statistical test.

11.3.7.2 Augmented Dickey-Fuller Test

The **ADF test** generalizes the DF test by allowing for:

1. **Lags of** Δy_t (to correct for serial correlation).
2. **Time trends** (to handle deterministic trends).

Regression Equation

$$\Delta y_t = \alpha + \theta y_{t-1} + \delta t + \gamma_1 \Delta y_{t-1} + \cdots + \gamma_p \Delta y_{t-p} + v_t$$

where $\theta = 1 - \rho$.

Hypotheses

- $H_0 : \theta = 0$ (Unit root: non-stationary)
- $H_a : \theta < 0$ (Stationary)

11.3.8 Newey-West Standard Errors

Newey-West standard errors, also known as **Heteroskedasticity and Autocorrelation Consistent (HAC) estimators**, provide valid inference when errors exhibit both heteroskedasticity (i.e., A4 Homoskedasticity assumption is violated) and serial correlation. These standard errors adjust for dependence in the error structure, ensuring that hypothesis tests remain valid.

Key Features

- **Accounts for autocorrelation**: Handles time dependence in error terms.
- **Accounts for heteroskedasticity**: Allows for non-constant variance across observations.
- **Ensures positive semi-definiteness**: Downweights longer-lagged covariances to maintain mathematical validity.

The estimator is computed as:

$$\hat{B} = T^{-1} \sum_{t=1}^{T} e_t^2 \mathbf{x_t'} \mathbf{x_t} + \sum_{h=1}^{g} \left(1 - \frac{h}{g+1}\right) T^{-1} \sum_{t=h+1}^{T} e_t e_{t-h} (\mathbf{x_t'} \mathbf{x_{t-h}} + \mathbf{x_{t-h}'} \mathbf{x_t})$$

where:

- T is the sample size,
- g is the chosen **lag truncation parameter** (bandwidth),
- e_t are the residuals from the OLS regression,
- $\mathbf{x}_t$ are the explanatory variables.

Choosing the Lag Length (g)

Selecting an appropriate lag truncation parameter (g) is crucial for balancing **efficiency** and **bias**. Common guidelines include:

- **Yearly data**: $g = 1$ or 2 usually suffices.
- **Quarterly data**: $g = 4$ or 8 accounts for seasonal dependencies.
- **Monthly data**: $g = 12$ or 14 captures typical cyclical effects.

Alternatively, data-driven methods can be used:

- **Newey-West Rule**: $g = \lfloor 4(T/100)^{2/9} \rfloor$
- **Alternative Heuristic**: $g = \lfloor T^{1/4} \rfloor$

```r
# Load necessary libraries
library(sandwich)
library(lmtest)

# Simulate data
set.seed(42)
T <- 100  # Sample size
time <- 1:T
x <- rnorm(T)
epsilon <- arima.sim(n = T, list(ar = 0.5))  # Autocorrelated errors
y <- 2 + 3 * x + epsilon  # True model

# Estimate OLS model
model <- lm(y ~ x)

# Compute Newey-West standard errors
lag_length <- floor(4 * (T / 100) ^ (2 / 9))  # Newey-West rule
nw_se <- NeweyWest(model, lag = lag_length, prewhite = FALSE)

# Display robust standard errors
coeftest(model, vcov = nw_se)
#>
#> t test of coefficients:
#>
#>             Estimate Std. Error t value  Pr(>|t|)
#> (Intercept)  1.71372    0.13189  12.993 < 2.2e-16 ***
```

```
#> x            3.15831    0.13402  23.567 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

11.3.8.1 Testing for Serial Correlation

Serial correlation (also known as **autocorrelation**) occurs when error terms are correlated across time:

$$E(\epsilon_t \epsilon_{t-h}) \neq 0 \quad \text{for some } h \neq 0$$

Steps for Detecting Serial Correlation

1. **Estimate an OLS regression**:
   - Run the regression of y_t on $\mathbf{x}_t$ and obtain residuals e_t.
2. **Test for autocorrelation in residuals**:
   - Regress e_t on $\mathbf{x}_t$ and its lagged residual e_{t-1}:

$$e_t = \gamma_0 + \mathbf{x}_t' \gamma + \rho e_{t-1} + v_t$$

   - Test whether ρ is significantly different from zero.
3. **Decision Rule**:
   - If ρ is statistically significant at the 5% level, reject the null hypothesis of no serial correlation.

Higher-Order Serial Correlation

To test for **higher-order autocorrelation**, extend the previous regression:

$$e_t = \gamma_0 + \mathbf{x}_t' \gamma + \rho_1 e_{t-1} + \rho_2 e_{t-2} + \cdots + \rho_p e_{t-p} + v_t$$

- **Jointly test** $\rho_1 = \rho_2 = \cdots = \rho_p = 0$ using an **F-test**.
- If the null is rejected, autocorrelation of order p is present.

Step 1: Estimate an OLS Regression and Obtain Residuals

```r
# Load necessary libraries
library(lmtest)
library(sandwich)

# Generate some example data
set.seed(123)
n <- 100
x <- rnorm(n)
y <- 1 + 0.5 * x + rnorm(n)  # True model: y = 1 + 0.5*x + e
```

```r
# Estimate the OLS regression
model <- lm(y ~ x)

# Obtain residuals
residuals <- resid(model)
```

Step 2: Test for Autocorrelation in Residuals

```r
# Create lagged residuals
lagged_residuals <- c(NA, residuals[-length(residuals)])

# Regress residuals on x and lagged residuals
autocorr_test_model <- lm(residuals ~ x + lagged_residuals)

# Summary of the regression
summary(autocorr_test_model)
#>
#> Call:
#> lm(formula = residuals ~ x + lagged_residuals)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -1.94809 -0.72539 -0.08105  0.58503  3.12941
#>
#> Coefficients:
#>                   Estimate Std. Error t value Pr(>|t|)
#> (Intercept)       0.008175   0.098112   0.083    0.934
#> x                -0.002841   0.107167  -0.027    0.979
#> lagged_residuals -0.127605   0.101746  -1.254    0.213
#>
#> Residual standard error: 0.9707 on 96 degrees of freedom
#>   (1 observation deleted due to missingness)
#> Multiple R-squared:  0.01614,    Adjusted R-squared:  -0.004354
#> F-statistic: 0.7876 on 2 and 96 DF,  p-value: 0.4579

# Test if the coefficient of lagged_residuals is significant
rho <- coef(autocorr_test_model)["lagged_residuals"]
rho_p_value <-
    summary(autocorr_test_model)$coefficients["lagged_residuals", "Pr(>|t|)"]

# Decision Rule
if (rho_p_value < 0.05) {
    cat("Reject the null hypothesis: There is evidence of serial correlation.\n")
} else {
    cat("Fail to reject the null hypothesis: No evidence of serial correlation.\n")
}
```

```
#> Fail to reject the null hypothesis: No evidence of serial correlation.
```

Step 3: Testing for Higher-Order Serial Correlation

```
# Number of lags to test
p <- 2  # Example: testing for 2nd order autocorrelation

# Create a matrix of lagged residuals
lagged_residuals_matrix <- sapply(1:p, function(i) c(rep(NA, i), residuals[1:(n - i)]))

# Regress residuals on x and lagged residuals
higher_order_autocorr_test_model <- lm(residuals ~ x + lagged_residuals_matrix)

# Summary of the regression
summary(higher_order_autocorr_test_model)
#>
#> Call:
#> lm(formula = residuals ~ x + lagged_residuals_matrix)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -1.9401 -0.7290 -0.1036  0.6359  3.0253
#>
#> Coefficients:
#>                          Estimate Std. Error t value Pr(>|t|)
#> (Intercept)              0.006263   0.099104   0.063    0.950
#> x                        0.010442   0.108370   0.096    0.923
#> lagged_residuals_matrix1 -0.140426   0.103419  -1.358    0.178
#> lagged_residuals_matrix2 -0.107385   0.103922  -1.033    0.304
#>
#> Residual standard error: 0.975 on 94 degrees of freedom
#>   (2 observations deleted due to missingness)
#> Multiple R-squared:  0.02667,    Adjusted R-squared:  -0.004391
#> F-statistic: 0.8587 on 3 and 94 DF,  p-value: 0.4655

# Joint F-test for the significance of lagged residuals
f_test <- car::linearHypothesis(higher_order_autocorr_test_model,
                           paste0("lagged_residuals_matrix", 1:p, " = 0"))

# Print the F-test results
print(f_test)
#> Linear hypothesis test
#>
#> Hypothesis:
#> lagged_residuals_matrix1 = 0
#> lagged_residuals_matrix2 = 0
```

```
#>
#> Model 1: restricted model
#> Model 2: residuals ~ x + lagged_residuals_matrix
#>
#>   Res.Df    RSS Df Sum of Sq      F Pr(>F)
#> 1     96 91.816
#> 2     94 89.368  2    2.4479 1.2874 0.2808

# Decision Rule
if (f_test$`Pr(>F)`[2] < 0.05) {
  cat("Reject the null hypothesis: There is evidence of higher-order serial correlation.\n")
} else {
  cat("Fail to reject the null hypothesis: No evidence of higher-order serial correlation.\n")
}
#> Fail to reject the null hypothesis: No evidence of higher-order serial correlation.
```

Corrections for Serial Correlation

If serial correlation is detected, the following adjustments should be made:

Problem	Solution
Mild serial correlation	Use Newey-West standard errors
Severe serial correlation	Use Generalized Least Squares or Prais-Winsten transformation
Autoregressive structure in errors	Model as an **ARMA** process
Higher-order serial correlation	Include lags of dependent variable or use HAC estimators with higher lag orders

---

11.4 Repeated Cross-Sectional Data

Repeated cross-sectional data consists of **multiple independent cross-sections** collected at different points in time. Unlike panel data, where the same individuals are tracked over time, repeated cross-sections **draw a fresh sample in each wave**.

This approach allows researchers to analyze **aggregate trends over time**, but it does **not track individual-level changes**.

Examples

- **General Social Survey (GSS) (U.S.)** – Conducted every two years with a new sample of respondents.
- **Political Opinion Polls** – Monthly voter surveys to track shifts in public sentiment.
- **National Health Surveys** – Annual studies with fresh samples to monitor **population-wide** health trends.
- **Educational Surveys** – Sampling different groups of students each year to assess learning outcomes.

11.4.1 Key Characteristics

1. **Fresh Sample in Each Wave**
   - Each survey represents an **independent** cross-section.
   - No respondent is tracked across waves.
2. **Population-Level Trends Over Time**
   - Researchers can study **how the distribution of characteristics (e.g., income, attitudes, behaviors) changes over time**.
   - However, individual trajectories **cannot** be observed.
3. **Sample Design Consistency**
   - To ensure comparability across waves, researchers must maintain **consistent**:
     - Sampling methods
     - Questionnaire design
     - Definitions of key variables

11.4.2 Statistical Modeling for Repeated Cross-Sections

Since repeated cross-sections do not track the same individuals, specific regression methods are used to **analyze changes over time**.

1. **Pooled Cross-Sectional Regression (Time Fixed Effects)**

Combines multiple survey waves into a single dataset while controlling for time effects:

$$y_i = \mathbf{x}_i\beta + \delta_1 y_1 + ... + \delta_T y_T + \epsilon_i$$

where:

- y_i is the outcome for individual i,
- $\mathbf{x}_i$ are explanatory variables,
- y_t are time period dummies,

- δ_t captures the **average change** in outcomes across time periods.

Key Features:

- Allows for different intercepts across time periods, capturing shifts in baseline outcomes.
- Tracks overall population trends without assuming a constant effect of $\mathbf{x}_i$ over time.

---

2. **Allowing for Structural Change in Pooled Cross-Sections (Time-Dependent Effects)**

To test whether relationships between variables change over time (**structural breaks**), interactions between time dummies and explanatory variables can be introduced:

$$y_i = \mathbf{x}_i\beta + \mathbf{x}_i y_1\gamma_1 + ... + \mathbf{x}_i y_T\gamma_T + \delta_1 y_1 + ... + \delta_T y_T + \epsilon_i$$

- **Interacting** x_i with time period dummies allows for:
  - **Different slopes** for each time period.
  - **Time-dependent effects** of explanatory variables.

Practical Application:

- If $\mathbf{x}_i$ represents education level and y_t represents survey year, an interaction term can test whether the effect of education on income has changed over time.
- Structural break tests help determine whether such time-varying effects are statistically significant.
- Useful for **policy analysis**, where a policy might impact certain subgroups differently across time.

---

3. **Difference-in-Means Over Time**

A simple approach to comparing **aggregate trends**:

$$\bar{y}_t - \bar{y}_{t-1}$$

- Measures whether the average outcome has changed over time.
- Common in policy evaluations (e.g., assessing the effect of minimum wage increases on average income).

---

4. **Synthetic Cohort Analysis**

Since repeated cross-sections do not track individuals, a **synthetic cohort** can be created by grouping observations based on shared characteristics:

- Example: If education levels are collected over multiple waves, we can track **average income changes** within education groups to approximate trends.

11.4.3 Advantages of Repeated Cross-Sectional Data

Advantage	Explanation
Tracks population trends	Useful for studying shifts in demographics, attitudes, and economic conditions over time.
Lower cost than panel data	Tracking individuals across multiple waves (as in panel studies) is expensive and prone to attrition.
No attrition bias	Unlike panel surveys, where respondents drop out over time, each wave draws a new representative sample.
Easier implementation	Organizations can design a single survey protocol and repeat it at set intervals without managing panel retention.

11.4.4 Disadvantages of Repeated Cross-Sectional Data

Disadvantage	Explanation
No individual-level transitions	Cannot track **how specific individuals change** over time (e.g., income mobility, changes in attitudes).
Limited causal inference	Since we observe different people in each wave, we cannot directly infer individual **cause-and-effect** relationships.
Comparability issues	Small differences in survey design (e.g., question wording or sampling frame) can make it difficult to compare across waves.

To ensure valid comparisons across time:

- **Consistent Sampling**: Each wave should use the same **sampling frame and methodology**.
- **Standardized Questions**: Small variations in **question wording** can introduce inconsistencies.

- **Weighting Adjustments**: If sampling strategies change, apply **survey weights** to maintain representativeness.
- **Accounting for Structural Changes**: Economic, demographic, or social changes may impact comparability.

11.5 Panel Data

Panel data (also called **longitudinal data**) consists of **observations of the same entities over multiple time periods**. Unlike repeated cross-sections, where new samples are drawn in each wave, panel data **tracks the same individuals, households, firms, or regions over time**, enabling richer statistical analysis.

Panel data combines **cross-sectional variation (differences across entities)** and **time-series variation (changes within entities over time)**.

Examples

- **Panel Study of Income Dynamics** – Follows households annually, collecting data on income, employment, and expenditures.
- **Medical Longitudinal Studies** – Tracks the same patients over months or years to study disease progression.
- **Firm-Level Financial Data** – Follows a set of companies over multiple years through financial statements.
- **Student Achievement Studies** – Follows the same students across different grade levels to assess academic progress.

Structure

- N entities (individuals, firms, etc.) observed over T time periods.
- The dataset can be:
  - **Balanced Panel**: All entities are observed in every time period.
  - **Unbalanced Panel**: Some entities have missing observations for certain periods.

Types of Panels

- **Short Panel**: Many individuals (N) but few time periods (T).
- **Long Panel**: Many time periods (T) but few individuals (N).
- **Both Large**: Large N and T (e.g., firm-level data over decades).

11.5.1 Advantages of Panel Data

Advantage	Explanation
Captures individual trajectories	Allows for studying how individuals or firms evolve over time.
Controls for unobserved heterogeneity	Fixed effects models remove time-invariant individual characteristics.
Stronger causal inference	Difference-in-differences and FE models improve causal interpretation.
More efficient estimates	Exploits both cross-sectional and time-series variation.

11.5.2 Disadvantages of Panel Data

Disadvantage	Explanation
Higher cost and complexity	Tracking individuals over time is resource-intensive.
Attrition bias	If certain individuals drop out systematically, results may be biased.
Measurement errors	Errors accumulate over time, leading to potential biases.

11.5.3 Sources of Variation in Panel Data

Since we observe **both individuals and time periods**, we distinguish three types of variation:

- **Overall variation**: Differences across both time and individuals.
- **Between variation**: Differences **between** individuals (cross-sectional variation).
- **Within variation**: Differences **within** individuals (time variation).

Estimate	Formula
Individual mean	$\bar{x}_i = \frac{1}{T}\sum_t x_{it}$
Overall mean	$\bar{x} = \frac{1}{NT}\sum_i \sum_t x_{it}$
Overall variance	$s_O^2 = \frac{1}{NT-1}\sum_i \sum_t (x_{it} - \bar{x})^2$
Between variance	$s_B^2 = \frac{1}{N-1}\sum_i (\bar{x}_i - \bar{x})^2$
Within variance	$s_W^2 = \frac{1}{NT-1}\sum_i \sum_t (x_{it} - \bar{x}_i)^2$

Note: $s_O^2 \approx s_B^2 + s_W^2$

11.5.4 Pooled OLS Estimator

The Pooled Ordinary Least Squares estimator is the simplest way to estimate relationships in panel data. It treats panel data as a **large cross-sectional dataset**, ignoring individual-specific effects and time dependence.

The pooled OLS model is specified as:

$$y_{it} = \mathbf{x}_{it}\beta + \epsilon_{it}$$

where:

- y_{it} is the dependent variable for individual i at time t,
- $\mathbf{x}_{it}$ is a vector of explanatory variables,
- β is the vector of coefficients to be estimated,
- $\epsilon_{it} = c_i + u_{it}$ is the composite error term.
  - c_i is the unobserved individual heterogeneity.
  - u_{it} is the idiosyncratic shock.

By treating all observations as **independent**, pooled OLS **assumes no systematic differences across individuals** beyond what is captured by $\mathbf{x}_{it}$.

For **pooled OLS** to be **consistent and unbiased**, the following conditions must hold:

1. Linearity in Parameters (A1)
   - The relationship between y_{it} and $\mathbf{x}_{it}$ is correctly specified as linear.
2. Full Rank Condition (A2)
   - The regressors are not perfectly collinear across individuals and time.
3. Strict Exogeneity (A3)
   - No correlation between regressors and error terms:

     $$E(\epsilon_{it}|\mathbf{x}_{it}) = 0$$

   - This ensures that OLS remains unbiased.
4. Homoskedasticity (A4)
   - Constant variance of errors:

     $$Var(\epsilon_{it}|\mathbf{x}_{it}) = \sigma^2$$

   - If violated (heteroskedasticity exists), standard errors must be adjusted using clustered robust approach, but OLS is still consistent.
5. No Autocorrelation Across Time (A5)

- The error term should not be correlated over time for a given individual:
$$E(\epsilon_{it}, \epsilon_{is}) = 0, \quad \forall t \neq s$$
- If this assumption fails, clustered standard errors are needed.

6. Random Sampling (A6)
   - Observations are independent across individuals:
$$(y_{i1}, ..., y_{iT}, x_{i1}, ..., x_{iT}) \perp (y_{j1}, ..., y_{jT}, x_{j1}, ..., x_{jT}) \quad \forall i \neq j$$
   - This assumption is often reasonable but not always valid (e.g., in firm-level or country-level panel data).

For **pooled OLS to be consistent**, we require A3a:

$$E(\mathbf{x}'_{it}(c_i + u_{it})) = 0$$

which holds if and only if:

1. **Exogeneity of** u_{it} (Time-varying error):
$$E(\mathbf{x}'_{it} u_{it}) = 0$$
   - Ensures that regressors are **not correlated** with the random error component.
2. **Random Effects Assumption** (Time-invariant error):
$$E(\mathbf{x}'_{it} c_i) = 0$$
   - Ensures that **unobserved heterogeneity** (c_i) is uncorrelated with regressors. If this assumption fails, pooled OLS suffers from **omitted variable bias**.

Implication:

- If c_i is correlated with $\mathbf{x}_{it}$, pooled OLS is **biased and inconsistent**.
- If c_i is uncorrelated with $\mathbf{x}_{it}$, pooled OLS is **consistent but inefficient**.

Variance Decomposition in Panel Data

Since panel data contains both **between-entity** and **within-entity** variation, the total variance can be decomposed into:

$$s_O^2 = s_B^2 + s_W^2$$

where:

- s_O^2 = **Overall variance** (variation over time and across individuals),
- s_B^2 = **Between variance** (variation between individuals),
- s_W^2 = **Within variance** (variation within individuals over time).

Key Insight:

- Pooled OLS does not separate within-individual and between-individual variation.
- Fixed Effects models eliminate c_i and use only within-individual variation.
- Random Effects models use both within and between variation.

Robust Inference in Pooled OLS

If the standard assumptions fail, adjustments are necessary:

1. **Heteroskedasticity:** If A4 (Homoskedasticity) is violated, standard errors must be adjusted using:
   1. White's Robust Standard Errors (for cross-sectional heteroskedasticity).
   2. Cluster-Robust Standard Errors (for panel-specific heteroskedasticity).
2. **Serial Correlation (Autocorrelation):** If errors are correlated across time:
   1. Use Newey-West Standard Errors for time dependence.
   2. Use Clustered Standard Errors at the Individual Level.
3. **Multicollinearity:** If regressors are highly correlated:
   1. Remove redundant variables.
   2. Use Variance Inflation Factor diagnostics.

Comparing Pooled OLS with Alternative Panel Models

Model	Assumption about c_i	Uses Within Variation?	Uses Between Variation?	Best When
Pooled OLS	Assumes c_i is uncorrelated with x_{it}	Yes	Yes	No individual heterogeneity
Fixed Effects	Removes c_i via demeaning	Yes	No	c_i is correlated with x_{it}
Random Effects	Assumes c_i is uncorrelated with x_{it}	Yes	Yes	c_i is uncorrelated with x_{it}

When to Use Pooled OLS?

- If individual heterogeneity is negligible
- If panel is short (T is small) and cross-section is large (N is big)
- If random effects assumption holds ($E(\mathbf{x}_{it}'c_i) = 0$)

If these conditions fail, **Fixed Effects or Random Effects models** should be used instead.

11.5.5 Individual-Specific Effects Model

In panel data, unobserved heterogeneity can arise when **individual-specific factors** (c_i) influence the dependent variable. These effects can be:

- **Correlated with regressors** ($E(\mathbf{x}_{it}'c_i) \neq 0$): Use the Fixed Effects estimator.
- **Uncorrelated with regressors** ($E(\mathbf{x}_{it}'c_i) = 0$): Use the Random Effects estimator.

The general model is:

$$y_{it} = \mathbf{x}_{it}\beta + c_i + u_{it}$$

where:

- c_i is the **individual-specific effect** (time-invariant),
- u_{it} is the **idiosyncratic error** (time-variant).

Comparing Fixed Effects and Random Effects

Model	Assumption on c_i	Uses Within Variation?	Uses Between Variation?	Best When
Fixed Effects	c_i is correlated with x_{it}	Yes	No	Unobserved heterogeneity bias present
Random Effects	c_i is uncorrelated with x_{it}	Yes	Yes	No correlation with regressors

---

11.5.6 Random Effects Estimator

The Random Effects (RE) estimator is a Feasible Generalized Least Squares method used in panel data analysis. It assumes that individual-specific effects (c_i) are uncorrelated with the explanatory variables ($\mathbf{x}_{it}$), allowing for estimation using both within-group (time variation) and between-group (cross-sectional variation).

The standard **Random Effects model** is:

$$y_{it} = \mathbf{x}_{it}\beta + c_i + u_{it}$$

where:

- y_{it} is the dependent variable for entity i at time t,
- $\mathbf{x}_{it}$ is a vector of explanatory variables,
- β represents the coefficients of interest,
- c_i is the **unobserved individual-specific effect** (time-invariant),
- u_{it} is the **idiosyncratic error** (time-varying).

In contrast to the Fixed Effects model, which eliminates c_i by demeaning the data, the Random Effects model treats c_i as a random variable and incorporates it into the error structure.

11.5.6.1 Key Assumptions for Random Effects

For the Random Effects estimator to be **consistent**, the following assumptions must hold:

1. **Exogeneity of the Time-Varying Error** (u_{it}) (A3a)

The idiosyncratic error term (u_{it}) must be **uncorrelated with regressors**:

$$E(\mathbf{x}'_{it}u_{it}) = 0$$

This assumption ensures that within-period variation in the regressors does not systematically affect the error term.

2. **Exogeneity of Individual-Specific Effects** (c_i) (A3a)

A crucial assumption of the RE model is that the individual effect (c_i) is uncorrelated with the explanatory variables:

$$E(\mathbf{x}'_{it}c_i) = 0$$

This means that the individual-specific unobserved characteristics do not systematically affect the choice of explanatory variables.

- **If this assumption fails**, the RE model produces **biased and inconsistent estimates** due to omitted variable bias.
- **If this assumption holds**, RE is **more efficient than FE** because it retains **both within-group and between-group variation**.

3. **No Serial Correlation in u_{it}**

The error term (u_{it}) must be **uncorrelated across time**:

$$E(u_{it}u_{is}) = 0, \quad \forall t \neq s$$

If this assumption fails:

- Standard errors will be incorrect.
- Generalized Least Squares adjustments or cluster-robust standard errors are required.

11.5.6.2 Efficiency of Random Effects

The RE estimator is a GLS estimator, meaning it is BLUE (Best Linear Unbiased Estimator) under homoskedasticity.

Scenario	**Efficiency of RE**
A4 (Homoskedasticity) holds	RE is the most efficient estimator.
A4 fails (Heteroskedasticity or Serial Correlation)	RE remains more efficient than Pooled OLS but is no longer optimal.

- When the variance of errors differs across individuals, RE can still be used but must be adjusted with robust standard errors.
- If errors are correlated over time, standard Newey-West or cluster-robust standard errors should be applied.

To efficiently estimate β, we transform the RE model using Generalized Least Squares.

Define the **quasi-demeaned transformation**:

$$\tilde{y}_{it} = y_{it} - \theta \bar{y}_i$$

$$\tilde{\mathbf{x}}_{it} = \mathbf{x}_{it} - \theta \bar{\mathbf{x}}_i$$

where:

$$\theta = 1 - \sqrt{\frac{\sigma_u^2}{T\sigma_c^2 + \sigma_u^2}}$$

- If $\theta = 1$, RE becomes the FE estimator.
- If $\theta = 0$, RE becomes Pooled OLS.

The final RE regression equation is:

$$\tilde{y}_{it} = \tilde{\mathbf{x}}_{it}\beta + \tilde{u}_{it}$$

which is estimated using GLS.

11.5.7 Fixed Effects Estimator

Also known as the **Within Estimator**, the **FE model** controls for individual-specific effects by removing them through transformation.

Key Assumption

If the RE assumption fails ($E(\mathbf{x}_{it}' c_i) \neq 0$), then:

- **Pooled OLS and RE become biased and inconsistent** (due to omitted variable bias).
- **FE is still consistent** because it eliminates c_i.

However, FE only corrects bias from time-invariant factors and does not handle time-variant omitted variables.

Challenges with FE

- **Bias with Lagged Dependent Variables**: FE is biased in dynamic models (Nickell, 1981; Narayanan and Nair, 2013).
- **Exacerbates Measurement Error**: FE can worsen **errors-in-variables bias**.

11.5.7.1 Demean (Within) Transformation

To remove c_i, we take the individual mean of the regression equation:

$$y_{it} = \mathbf{x}_{it}\beta + c_i + u_{it}$$

Averaging over time (T):

$$\bar{y}_i = \bar{\mathbf{x}}_i\beta + c_i + \bar{u}_i$$

Subtracting the second equation from the first (i.e., **within transformation**):

$$(y_{it} - \bar{y}_i) = (\mathbf{x}_{it} - \bar{\mathbf{x}}_i)\beta + (u_{it} - \bar{u}_i)$$

This transformation:

- Eliminates c_i, solving omitted variable bias.
- Only uses within-individual variation.

The transformed regression is estimated via OLS:

$$y_{it} - \bar{y}_i = (\mathbf{x}_{it} - \bar{\mathbf{x}}_i)\beta + d_1\delta_1 + \cdots + d_{T-2}\delta_{T-2} + (u_{it} - \bar{u}_i)$$

where

- d_t is a time dummy variable, which equals 1 if the observation in the time periods t, and 0 otherwise. This variable is for period $t = 1, \ldots, T-1$ (one period omitted to avoid perfect multicollinearity).
- δ_t is the coefficient on the time dummy, capturing aggregate shocks that affect all individual in period t.

Key Conditions for Consistency:

- Strict Exogeneity (A3):

$$E[(\mathbf{x}_{it} - \bar{\mathbf{x}}_i)'(u_{it} - \bar{u}_i)] = 0$$

- Time-invariant variables are dropped (e.g., gender, ethnicity). If you're interested in the effect of these time-invariant variables, consider using either OLS or the between estimator.
- Cluster-Robust Standard Errors should be used.

11.5.7.2 Dummy Variable Approach

The Dummy Variable Approach is an alternative way to estimate Fixed Effects in panel data. Instead of transforming the data by demeaning (Within Transformation), this method explicitly includes individual dummy variables to control for entity-specific heterogeneity.

The general FE model is:

$$y_{it} = \mathbf{x}_{it}\beta + c_i + u_{it}$$

where:

- c_i is the unobserved, **time-invariant individual effect** (e.g., ability, cultural preferences, managerial style).
- u_{it} is the **idiosyncratic error term** (fluctuates over time and across individuals).

To estimate this model using the Dummy Variable Approach, we include a separate dummy variable for each individual:

$$y_{it} = \mathbf{x}_{it}\beta + d_1\delta_1 + ... + d_{T-2}\delta_{T-2} + c_1\gamma_1 + ... + c_{n-1}\gamma_{n-1} + u_{it}$$

where:

- c_i is now modeled explicitly as a dummy variable ($c_i\gamma_i$) for each individual.
- d_t are time dummies, capturing time-specific shocks.
- δ_t are coefficients on time dummies, controlling for common time effects.

Interpretation of the Dummy Variables

- The dummy variable c_i takes a value of 1 for individual i and 0 otherwise:

$$c_i = \begin{cases} 1 & \text{if observation is for individual } i \\ 0 & \text{otherwise} \end{cases}$$

- These N dummy variables absorb all individual-specific variation, ensuring that only within-individual (over-time) variation remains.

Advantages of the Dummy Variable Approach

1. Easy to Interpret: Explicitly includes entity-specific effects, making it easier to understand how individual heterogeneity is modeled.
2. Equivalent to the Within (Demean) Transformation: Mathematically, this approach produces the same coefficient estimates as the Within Transformation.
3. Allows for Inclusion of Time Dummies: The model can easily incorporate time dummies (d_t) to control for period-specific shocks.

Limitations of the Dummy Variable Approach

1. **Computational Complexity with Large N**

- Adding N dummy variables significantly increases the number of parameters estimated.
- If N is very large (e.g., 10,000 individuals), this approach can be computationally expensive.

2. **Standard Errors Are Incorrectly Estimated**
   - The standard errors for c_i dummy variables are often incorrectly calculated, as they absorb all within-individual variation.
   - This is why the Within Transformation (Demeaning Approach) is generally preferred.
3. **Consumes Degrees of Freedom**
   - Introducing N additional parameters reduces degrees of freedom, which can lead to overfitting.

11.5.7.3 First-Difference Approach

An alternative way to eliminate individual-specific effects (c_i) is to **take first differences across time**, rather than subtracting the individual mean.

The FE model:

$$y_{it} = \mathbf{x}_{it}\beta + c_i + u_{it}$$

Since c_i is constant over time, taking the first difference:

$$y_{it} - y_{i(t-1)} = (\mathbf{x}_{it} - \mathbf{x}_{i(t-1)})\beta + (u_{it} - u_{i(t-1)})$$

This transformation **removes** c_i completely, leaving a model that can be estimated using Pooled OLS.

Advantages of the First-Difference Approach

1. **Eliminates Individual Effects** (c_i)
   - Since c_i is time-invariant, differencing removes it from the equation.
2. **Works Well with Few Time Periods (**T is Small)
   - If T is small, first-differencing is often preferred over the Within Transformation, as it does not require averaging over many periods.
3. **Less Computationally Intensive**
   - Unlike the Dummy Variable Approach, which requires estimating N additional parameters, the First-Difference Approach reduces the dimensionality of the problem.

Limitations of the First-Difference Approach

1. **Cannot Handle Missing Observations Well**
   - If data is missing in period $t-1$ for an individual, then the corresponding first-difference observation is lost.
   - This can significantly reduce sample size in unbalanced panels.
2. **Reduces Number of Observations by One**
   - Since first differences require $y_{i(t-1)}$ to exist, the model loses one time period ($T-1$ observations per individual instead of T).
3. **Can Introduce Serial Correlation**
   - Since we are differencing $u_{it} - u_{i(t-1)}$, the error term now exhibits autocorrelation.
   - This means standard OLS assumptions (independent errors) no longer hold, requiring the use of robust standard errors.

11.5.7.4 Comparison of FE Approaches

Approach	How it Works	Pros	Cons
Dummy Variable	Includes individual dummies (c_i) in regression	Intuitive, easy to interpret	Computationally expensive for large N, standard errors may be incorrect
Within Transformation (Demeaning)	Subtracts individual mean from each variable	Computationally efficient, correct standard errors	Cannot estimate time-invariant variables
First-Difference Approach	Takes time differences to remove c_i	Simple, works well for small T	Reduces sample size, introduces autocorrelation

---

Key Insights

1. **The Dummy Variable Approach explicitly models** c_i but is computationally expensive for large N.
2. **The Within (Demean) Transformation is the most commonly used FE method** because it is **computationally efficient** and **produces correct standard errors**.
3. **The First-Difference Approach is useful when** T is small, but it reduces sample size and introduces autocorrelation.
4. **If data has many missing values, First-Difference is not recommended** due to its sensitivity to gaps in observations.
5. **Time dummies** (d_t) can be included in any FE model to control for time shocks that affect all individuals.

6. FE only exploits **within variation**, meaning only **status changes contribute to** β estimates.
7. **With limited status changes, standard errors explode** (small number of switchers leads to high variance).
8. **Treatment effect is non-directional** but can be parameterized.
9. **Switchers vs. Non-Switchers**:
    - If **switchers differ fundamentally**, the FE estimator may still be biased.
    - Descriptive statistics on switchers/non-switchers help verify robustness.

11.5.7.5 Variance of Errors in FE

FE reduces variation by removing c_i, which affects **error variance**:

$$\hat{\sigma}_\epsilon^2 = \frac{SSR_{OLS}}{NT - K}$$

$$\hat{\sigma}_u^2 = \frac{SSR_{FE}}{NT - (N + K)} = \frac{SSR_{FE}}{N(T - 1) - K}$$

Implication:

- The variance of the error may increase or decrease because:
    - SSR can increase (since FE eliminates between variation).
    - Degrees of freedom decrease (as more parameters are estimated).

11.5.7.6 Fixed Effects Examples

11.5.7.6.1 Intergenerational Mobility – Blau (1999) Research Questions

- Does **transferring resources** to low-income families improve **upward mobility** for children?
- What are the **mechanisms** of intergenerational mobility?

Mechanisms for Intergenerational Mobility

There are multiple pathways through which **parental income influences child outcomes**:

1. Genetics (Ability Endowment)
    - If mobility is purely genetic, policy cannot affect outcomes.
2. Environmental Indirect Effects
    - Family background, peer influences, school quality.
3. Environmental Direct Effects

- Parental investments in education, health, social capital.

4. Financial Transfers
   - Direct monetary support, inheritance, wealth accumulation.

One way to measure the impact of income on human capital accumulation is:

$$\frac{\%\Delta\text{Human Capital}}{\%\Delta\text{Parental Income}}$$

where human capital includes education, skills, and job market outcomes.

Income is measured in different ways to capture its **long-term effects**:

1. **Total household income**
2. **Wage income**
3. **Non-wage income**
4. **Annual vs. Permanent Income** (important distinction for long-term analysis)

Key control variables must be **exogenous** to avoid bias. Bad Controls are those that are **jointly determined with the dependent variable** (e.g., mother's labor force participation).

Exogenous controls:

- Mother's race
- Birth location
- Parental education
- Household structure at age 14

The estimated model is:

$$Y_{ijt} = X_{jt}\beta_i + I_{jt}\alpha_i + \epsilon_{ijt}$$

where:

- i = test (e.g., academic test score).
- j = individual (child).
- t = time.
- X_{jt} = observable child characteristics.
- I_{jt} = parental income.
- ϵ_{ijt} = error term.

Grandmother's Fixed-Effects Model

Since a child (j) is **nested** within a mother (m), and a mother is **nested** within a grandmother (g), we estimate:

$$Y_{ijgmt} = X_{it}\beta_i + I_{jt}\alpha_i + \gamma_g + u_{ijgmt}$$

where:

- g = Grandmother, m = Mother, j = Child, t = Time.
- γ_g captures both grandmother and mother fixed effects.
- The nested structure controls for genetic and fixed family environment effects.
- **Cluster standard errors** at the family level to **account for correlation in errors** across generations.

Pros of Grandmother FE Model

- Controls for genetics + fixed family background
- Allows estimation of income effects independent of family background

Cons

- Might not fully control for unobserved heterogeneity
- Measurement errors in income can exaggerate attenuation bias

---

11.5.7.6.2 Fixed Effects in Teacher Quality Studies – Babcock (2010)

The study investigates:

- How teacher quality influences student performance.
- Whether students adjust course selection behavior based on past grading experiences.
- How to properly estimate teacher fixed effects while addressing selection bias and measurement error.

The initial model estimates student performance (T_{ijct}) based on class expectations and student characteristics:

$$T_{ijct} = \alpha_0 + S_{jct}\alpha_1 + X_{ijct}\alpha_2 + u_{ijct}$$

where:

- T_{ijct} = Student test score.

- S_{jct} = Class-level grading expectation (e.g., expected GPA in the course).
- X_{ijct} = Individual student characteristics.
- i = Student, j = Instructor, c = Course, t = Time.
- u_{ijct} = Idiosyncratic error term.

A key issue in this model is that **grading expectations** may not be randomly assigned. If students select into courses based on grading expectations, **simultaneity bias** can arise.

To control for **instructor and course heterogeneity**, the model introduces **teacher-course fixed effects** (μ_{jc}):

$$T_{ijct} = \beta_0 + S_{jct}\beta_1 + X_{ijct}\beta_2 + \mu_{jc} + \epsilon_{ijct}$$

where:

- μ_{jc} is a unique fixed effect for each instructor-course combination.
- This controls for instructor-specific grading policies and course difficulty.
- It differs from a simple instructor effect (θ_j) and course effect (δ_c) because it captures interaction effects.

Implications of Instructor-Course Fixed Effects

1. **Reduces Bias from Course Shopping**
   - Students may select courses based on grading expectations.
   - Including μ_{jc} controls for the fact that some instructors systematically assign easier grades.
2. **Shifts in Student Expectations**
   - Even if course content remains constant, students adjust their expectations based on past grading experiences.
   - This influences their future course selection behavior.

Identification Strategy

A key challenge in estimating teacher effects is **endogeneity** from:

1. **Simultaneity Bias**
   1. Grading expectations (S_{jct}) and student performance may be jointly determined.
   2. If grading expectations are based on past student performance, **OLS will be biased**.
2. **Unobserved Teacher Characteristics**

- Some teachers may have innate ability to motivate students, leading to higher student performance independent of observable teacher traits.

To address these concerns, the model first controls for **observable teacher characteristics**:

$$\begin{aligned} Y_{ijt} = X_{it}\beta_1 + \text{Teacher Experience}_{jt}\beta_2 + \text{Teacher Education}_{jt}\beta_3 \\ + \text{Teacher Score}_{it}\beta_4 + \cdots + \epsilon_{ijt} \end{aligned}$$

However, if **teacher characteristics are correlated with unobserved ability**, we replace them with **teacher fixed effects**:

$$Y_{ijt} = X_{it}\alpha + \Gamma_{it}\theta_j + u_{ijt}$$

where:

- θ_j = **Teacher Fixed Effect**, capturing all time-invariant teacher characteristics.
- Γ_{it} represents within-teacher variation.

To further analyze teacher impact, we express student test scores as:

$$Y_{ijt} = X_{it}\gamma + \epsilon_{ijt}$$

where:

- γ represents the between and within variation.
- e_{ijt} is the prediction error.

Decomposing the error term:

$$e_{ijt} = T_{it}\delta_j + \tilde{e}_{ijt}$$

where:

- δ_j = Group-level teacher effect.
- $\tilde{e}_{ijt}$ = Residual error.

To control for **prior student performance**, we introduce **lagged test scores**:

$$Y_{ijkt} = Y_{ijkt-1} + X_{it}\beta + T_{it}\tau_j + (W_i + P_k + \epsilon_{ijkt})$$

where:

- Y_{ijkt-1} = Lagged student test score.
- τ_j = Teacher Fixed Effect.
- W_i = Student Fixed Effect.
- P_k = School Fixed Effect.
- $u_{ijkt} = W_i + P_k + \epsilon_{ijkt}$.

A major issue is **selection bias**:

- If students **sort into better teachers**, the teacher effect (τ) may be **overestimated**.
- Bias in τ for teacher j is:

$$\frac{1}{N_j}\sum_{i=1}^{N_j}(W_i + P_k + \epsilon_{ijkt})$$

- where N_j is the number of students in class with teacher j.
- **Smaller class sizes → Higher bias** in teacher effect estimates because $\frac{1}{N_j}\sum_{i=1}^{N_j}\epsilon_{ijkt} \neq 0$ will inflate the teacher fixed effect. If we use the random teacher effects instead, τ will still contain bias and we do not know the direction of the bias.

If teachers **switch schools**, we can separately estimate:

- Teacher Fixed Effects (τ_j)
- School Fixed Effects (P_k)

The **mobility web** refers to the network of teacher transitions across schools, which helps in identifying both teacher and school fixed effects.

- Thin mobility web: Few teachers switch schools, making it harder to separate teacher effects from school effects.
- Thick mobility web: Many teachers switch schools, improving identification of teacher quality independent of school characteristics.

The **panel data model** capturing student performance over time is:

$$Y_{ijkt} = Y_{ijk(t-1)}\alpha + X_{it}\beta + T_{it}\tau + P_k + \epsilon_{ijkt}$$

where:

- Y_{ijkt} = Student performance at time t.

- $Y_{ijk(t-1)}$ = Lagged student test score.
- X_{it} = Student characteristics.
- T_{it} = Teacher effect (τ).
- P_k = School fixed effect.
- ϵ_{ijkt} = Idiosyncratic error term.

If we apply fixed effects (demeaning transformation):

$$Y_{ijkt} - \bar{Y}_{ijk} = (X_{it} - \bar{X}_i)\beta + (T_{it} - \bar{T}_i)\tau + (P_k - \bar{P}) + (\epsilon_{ijkt} - \bar{\epsilon}_{ijk})$$

- This transformation removes teacher fixed effects (τ).
- If we want to explicitly estimate τ, we must include teacher fixed effects before demeaning.

The paper argues that controlling for school fixed effects (P_k) ensures no selection bias, meaning students are **randomly assigned** within schools.

A key claim in the paper is that teacher quality (τ) does not depend on the number of students per teacher (N_j).

To test this, we examine the variance of estimated teacher effects:

$$var(\tau)$$

If:

$$var(\tau) = 0$$

this implies **teacher quality does not impact student performance**.

To empirically test this, the study analyzes:

$$\frac{1}{N_j} \sum_{i=1}^{N_j} \epsilon_{ijkt}$$

which represents **teacher-level average residuals**.

Key Finding:

- The variance of teacher effects remains stable across different class sizes (N_j).
- This suggests that random assignment of students across teachers is not biasing τ.

Since teacher effects (τ_j) are **estimated with error** (Spin-off of Measurement Error), we decompose them as:

$$\hat{\tau}_j = \tau_j + \lambda_j$$

where:

- τ_j = True teacher effect.
- λ_j = Measurement error (e.g., sampling error, estimation noise).

Assuming τ_j and λ_j are uncorrelated:

$$cov(\tau_j, \lambda_j) = 0$$

this means that **the randomness in student assignments does not systematically bias teacher quality estimates**.

The total observed variance in estimated teacher effects is:

$$var(\hat{\tau}) = var(\tau) + var(\lambda)$$

Rearranging:

$$var(\tau) = var(\hat{\tau}) - var(\lambda)$$

Since we observe $var(\hat{\tau})$, we need to estimate $var(\lambda)$.

Measurement error variance ($var(\lambda)$) can be approximated using the average squared standard error of teacher effects:

$$var(\lambda) = \frac{1}{J} \sum_{j=1}^{J} \hat{\sigma}_j^2$$

where $\hat{\sigma}_j^2$ is the squared standard error of teacher j (which depends on sample size N_j).

The **signal-to-noise ratio** (or **reliability**) of teacher effect estimates is:

$$\frac{var(\tau)}{var(\hat{\tau})} = \text{Reliability}$$

where:

- **Higher reliability** indicates that most of the variation comes from true teacher effects (τ) rather than noise.
- **Lower reliability** suggests that a large portion of variation is due to **measurement error**.

The proportion of **error variance** in estimated teacher effects is:

$$1 - \frac{var(\tau)}{var(\hat{\tau})} = \text{Noise}$$

Even if true teacher quality depends on class size (N_j), our method for estimating λ remains unaffected.

To check whether **teacher effects are biased by sampling error**, we regress estimated teacher effects ($\hat{\tau}_j$) on teacher characteristics (X_j):

$$\hat{\tau}_j = \beta_0 + X_j\beta_1 + \epsilon_j$$

If teacher characteristics **do not predict sampling error**, then:

$$R^2 \approx 0$$

This would confirm that **teacher characteristics are uncorrelated with measurement error**, validating the identification strategy.

11.5.8 Tests for Assumptions in Panel Data Analysis

We typically don't test heteroskedasticity explicitly because robust covariance matrix estimation is used. However, other key assumptions should be tested before choosing the appropriate panel model.

```
library("plm")
data("EmplUK", package="plm")
data("Produc", package="plm")
data("Grunfeld", package="plm")
data("Wages", package="plm")
```

11.5.8.1 Poolability Test

Tests whether coefficients are the same across individuals (also known as an F-test of stability or Chow test).

- H_0: All individuals have the same coefficients (i.e., equal coefficients for all individuals).
- H_a: Different individuals have different coefficients.

Notes:

- A fixed effects model assumes different intercepts per individual.
- A random effects model assumes a common intercept.

```
library(plm)
plm::pooltest(inv ~ value + capital,
              data = Grunfeld,
              model = "within")
#>
#>  F statistic
#>
#> data:  inv ~ value + capital
#> F = 5.7805, df1 = 18, df2 = 170, p-value = 1.219e-10
#> alternative hypothesis: unstability
```

If the null is rejected, we should not use a pooled OLS model.

11.5.8.2 Testing for Individual and Time Effects

Checks for the presence of individual or time effects, or both.

- **Types of tests:**
  - `honda`: Default test for individual effects (Honda, 1985)
  - `bp`: Breusch-Pagan test for unbalanced panels (Breusch and Pagan, 1980)
  - `kw`: King-Wu test for unbalanced panels with two-way effects (King and Wu, 1997)
  - `ghm`: Gourieroux, Holly, and Monfort test for two-way effects (Gourieroux et al., 1982)

```
pFtest(inv ~ value + capital,
       data = Grunfeld,
       effect = "twoways")
#>
#>  F test for twoways effects
#>
#> data:  inv ~ value + capital
#> F = 17.403, df1 = 28, df2 = 169, p-value < 2.2e-16
#> alternative hypothesis: significant effects
pFtest(inv ~ value + capital,
```

```
       data = Grunfeld,
       effect = "individual")
#>
#>  F test for individual effects
#>
#> data:  inv ~ value + capital
#> F = 49.177, df1 = 9, df2 = 188, p-value < 2.2e-16
#> alternative hypothesis: significant effects
pFtest(inv ~ value + capital,
       data = Grunfeld,
       effect = "time")
#>
#>  F test for time effects
#>
#> data:  inv ~ value + capital
#> F = 0.23451, df1 = 19, df2 = 178, p-value = 0.9997
#> alternative hypothesis: significant effects
```

If the null hypothesis is rejected, a fixed effects model is more appropriate.

11.5.8.3 Cross-Sectional Dependence (Contemporaneous Correlation)

Tests whether residuals across entities are correlated.

```
pcdtest(inv ~ value + capital,
        data = Grunfeld, model = "within")
#>
#>  Pesaran CD test for cross-sectional dependence in panels
#>
#> data:  inv ~ value + capital
#> z = 4.6612, p-value = 3.144e-06
#> alternative hypothesis: cross-sectional dependence
```

11.5.8.3.1 Global Cross-Sectional Dependence

11.5.8.3.2 Local Cross-Sectional Dependence Uses a spatial weight matrix **w**.

```
pcdtest(inv ~ value + capital,
        data = Grunfeld,
        model = "within",
        w = weight_matrix)
```

If the null is rejected, cross-sectional correlation exists and should be addressed.

11.5.8.4 Serial Correlation in Panel Data

- **Null hypothesis**: There is no serial correlation.
- Serial correlation is typically observed in macro panels with long time series (large N and T). It is less relevant in micro panels with short time series (small T and large N).
- **Sources of Serial Correlation:**
  - **Unobserved individual effects:** Time-invariant error components.
  - **Idiosyncratic error terms:** Often modeled as an autoregressive process (e.g., AR(1)).
  - Typically, "serial correlation" refers to the second type (idiosyncratic errors).

Types of Serial Correlation Tests

- **Marginal tests:** Test for one type of dependence at a time but may be biased towards rejection.
- **Joint tests:** Detect both sources of dependence but do not distinguish the source of the problem.
- **Conditional tests:** Assume one dependence structure is correctly specified and test for additional departures.

11.5.8.4.1 Unobserved Effects Test

- A semi-parametric test for unobserved effects, with the test statistic $W \sim N$ regardless of the error distribution.
- Null hypothesis (H_0): No unobserved effects ($\sigma_\mu^2 = 0$), which supports using pooled OLS.
- Under H_0: The covariance matrix of residuals is diagonal (no off-diagonal correlations).
- Robustness: The test is robust to both unobserved individual effects and serial correlation.

```r
library(plm)
data("Produc", package = "plm")

# Wooldridge test for unobserved individual effects
pwtest(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
       data = Produc)
#>
#>  Wooldridge's test for unobserved individual effects
#>
#> data:  formula
#> z = 3.9383, p-value = 8.207e-05
#> alternative hypothesis: unobserved effect
```

Interpretation: If we reject H_0, pooled OLS is inappropriate due to the presence of unobserved effects.

11.5.8.4.2 Locally Robust Tests for Serial Correlation and Random Effects

- Joint LM Test for Random Effects and Serial Correlation
- A Lagrange Multiplier test to jointly detect:
  - Random effects (panel-level variance components).
  - Serial correlation (time-series dependence).
- Null Hypothesis: Normality and homoskedasticity of idiosyncratic errors (Baltagi and Li, 1991, 1995).
  - This is equivalent to assuming there is no presence of serial correlation, and random effects.

```
# Baltagi and Li's joint test for serial correlation and random effects
pbsytest(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
         data = Produc,
         test = "j")
#>
#>  Baltagi and Li AR-RE joint test
#>
#> data:  formula
#> chisq = 4187.6, df = 2, p-value < 2.2e-16
#> alternative hypothesis: AR(1) errors or random effects
```

Interpretation: If we reject H_0, either serial correlation, random effects, or both are present. But we don't know the source of dependence.

To distinguish the source of dependence, we use either (both tests assume normality and homoskedasticity) (Bera et al., 2001):

BSY Test for Serial Correlation

```
pbsytest(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
         data = Produc)
#>
#>  Bera, Sosa-Escudero and Yoon locally robust test
#>
#> data:  formula
#> chisq = 52.636, df = 1, p-value = 4.015e-13
#> alternative hypothesis: AR(1) errors sub random effects
```

BSY Test for Random Effects

```
pbsytest(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
         data = Produc,
         test = "re")
#>
#>  Bera, Sosa-Escudero and Yoon locally robust test (one-sided)
```

```
#>
#> data:  formula
#> z = 57.914, p-value < 2.2e-16
#> alternative hypothesis: random effects sub AR(1) errors
```

If serial correlation is "known" to be absent (based on the BSY test), the **LM test for random effects** is superior.

```
plmtest(inv ~ value + capital,
        data = Grunfeld,
        type = "honda")
#>
#>  Lagrange Multiplier Test - (Honda)
#>
#> data:  inv ~ value + capital
#> normal = 28.252, p-value < 2.2e-16
#> alternative hypothesis: significant effects
```

If random effects are absent (based on the BSY test), we use **Breusch-Godfrey's serial correlation test** (Breusch, 1978; Godfrey, 1978).

```
lmtest::bgtest()
```

If Random Effects are Present: Use Baltagi and Li's Test

Baltagi and Li's test detects serial correlation in AR(1) and MA(1) processes under random effects.

- Null hypothesis (H_0): Uncorrelated errors.
- Note:
  - The test has power only against positive serial correlation (one-sided).
  - It is applicable only to balanced panels

```
pbltest(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
        data = Produc,
        alternative = "onesided")
#>
#>  Baltagi and Li one-sided LM test
#>
#> data:  log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp
#> z = 21.69, p-value < 2.2e-16
#> alternative hypothesis: AR(1)/MA(1) errors in RE panel model
```

11.5.8.4.3 General Serial Correlation Tests

- Applicable to **random effects, pooled OLS, and fixed effects** models.

- Can test for **higher-order serial correlation**.

```
# Baltagi-Griffin test for higher-order serial correlation
plm::pbgtest(plm::plm(inv ~ value + capital,
                      data = Grunfeld,
                      model = "within"),
             order = 2)
#>
#>  Breusch-Godfrey/Wooldridge test for serial correlation in panel models
#>
#> data:  inv ~ value + capital
#> chisq = 42.587, df = 2, p-value = 5.655e-10
#> alternative hypothesis: serial correlation in idiosyncratic errors
```

For short panels (Small T, Large N), use Wooldridge's test:

```
pwartest(log(emp) ~ log(wage) + log(capital),
         data = EmplUK)
#>
#>  Wooldridge's test for serial correlation in FE panels
#>
#> data:  plm.model
#> F = 312.3, df1 = 1, df2 = 889, p-value < 2.2e-16
#> alternative hypothesis: serial correlation
```

11.5.8.5 Unit Roots and Stationarity in Panel Data

11.5.8.5.1 Dickey-Fuller Test for Stochastic Trends

- Purpose: Tests for the presence of a unit root (non-stationarity) in a time series.
- Null hypothesis (H_0): The series is non-stationary (i.e., it has a unit root).
- Alternative hypothesis (H_A): The series is stationary (no unit root).
- Decision Rule:
  - If the test statistic is less than the critical value (or $p < 0.05$), reject H_0, indicating stationarity.
  - If the test statistic is greater than the critical value (or $p \geq 0.05$), fail to reject H_0, suggesting the presence of a unit root.

```
library(tseries)

# Example: Test for unit root in GDP data
adf.test(Produc$gsp, alternative = "stationary")
#>
#>  Augmented Dickey-Fuller Test
#>
#> data:  Produc$gsp
#> Dickey-Fuller = -6.5425, Lag order = 9, p-value = 0.01
```

```
#> alternative hypothesis: stationary
```

If we reject H_0, the series is **stationary** and does not exhibit a stochastic trend.

11.5.8.5.2 Levin-Lin-Chu Unit Root Test

- Purpose: Tests for the presence of a unit root in a panel dataset.
- Null hypothesis (H_0): The series has a unit root (non-stationary).
- Alternative hypothesis (H_A): The series is stationary.
- Assumptions: Requires large N (cross-sections) and moderate T (time periods).
- Decision Rule: If the test statistic is less than the critical value or $p < 0.05$, reject H_0 (evidence of stationarity).

```
library(tseries)
library(plm)

# Levin-Lin-Chu (LLC) Unit Root Test
purtest(Grunfeld, test = "levinlin")
#>
#>  Levin-Lin-Chu Unit-Root Test (ex. var.: None)
#>
#> data:  Grunfeld
#> z = 0.39906, p-value = 0.6551
#> alternative hypothesis: stationarity
```

If we reject H_0, the series is **stationary**.

11.5.8.6 Heteroskedasticity in Panel Data

11.5.8.6.1 Breusch-Pagan Test

- Purpose: Detects heteroskedasticity in regression residuals.
- Null hypothesis (H_0): The data is homoskedastic (constant variance).
- Alternative hypothesis (H_A): The data exhibits heteroskedasticity (non-constant variance).
- Decision Rule:
  - If the p-value is small (e.g., $p < 0.05$), reject H_0, suggesting heteroskedasticity.
  - If the p-value is large ($p \geq 0.05$), fail to reject H_0, implying homoskedasticity.

```
library(lmtest)

# Fit a panel model (pooled OLS)
```

```r
model <- lm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
            data = Produc)

# Breusch-Pagan Test for Heteroskedasticity
bptest(model)
#>
#>  studentized Breusch-Pagan test
#>
#> data:  model
#> BP = 80.033, df = 4, p-value < 2.2e-16
```

If heteroskedasticity is detected, we need to adjust for it using **robust standard errors**.

11.5.8.6.2 Robust Covariance Matrix Estimation (Sandwich Estimator) If heteroskedasticity is present, robust covariance matrix estimation is recommended. Different estimators apply depending on whether serial correlation is also an issue.

Choosing the Correct Robust Covariance Matrix Estimator

Estimator	Corrects for Heteroskedasticity?	Corrects for Serial Correlation?	Recommended For
`"white1"`	Yes	No	Random Effects
`"white2"`	Yes (common variance within groups)	No	Random Effects
`"arellano"`	Yes	Yes	Fixed Effects

```r
library(plm)

# Fit a fixed effects model
fe_model <- plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
                data = Produc,
                model = "within")

# Compute robust standard errors using Arellano's method
coeftest(fe_model, vcov = vcovHC(fe_model, method = "arellano"))
#>
#> t test of coefficients:
#>
#>             Estimate Std. Error t value  Pr(>|t|)
#> log(pcap) -0.0261497  0.0603262 -0.4335   0.66480
```

```
#> log(pc)     0.2920069  0.0617425  4.7294 2.681e-06 ***
#> log(emp)    0.7681595  0.0816652  9.4062 < 2.2e-16 ***
#> unemp      -0.0052977  0.0024958 -2.1226   0.03411 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Using a robust covariance matrix corrects for heteroskedasticity and/or serial correlation, ensuring valid inference.

11.5.9 Model Selection in Panel Data

Panel data models must be chosen based on the structure of the data and underlying assumptions. This section provides guidance on selecting between Pooled OLS, Random Effects, and Fixed Effects models.

11.5.9.1 Pooled OLS vs. Random Effects

The choice between POLS and RE depends on whether there are unobserved individual effects.

Breusch-Pagan Lagrange Multiplier Test

- **Purpose:** Tests whether a random effects model is preferable to a pooled OLS model.
- **Null hypothesis** (H_0): Variance across entities is zero (i.e., no panel effect $\rightarrow$ POLS is preferred).
- **Alternative hypothesis** (H_A): There is significant panel-level variation $\rightarrow$ **RE is preferable to POLS**.
- **Decision Rule:** If $p < 0.05$, reject H_0, indicating that RE is preferred.

```r
library(plm)

# Breusch-Pagan LM Test
plmtest(plm(inv ~ value + capital, data = Grunfeld,
            model = "pooling"), type = "bp")
#>
#>  Lagrange Multiplier Test - (Breusch-Pagan)
#>
#> data:  inv ~ value + capital
#> chisq = 798.16, df = 1, p-value < 2.2e-16
#> alternative hypothesis: significant effects
```

If the test is significant, RE is more appropriate than POLS.

11.5.9.2 Fixed Effects vs. Random Effects

The choice between **FE and RE** depends on whether the individual-specific effects are correlated with the regressors.

Key Assumptions and Properties

Hypothesis	If True
$H_0 : \text{Cov}(c_i, \mathbf{x_{it}}) = 0$	$\hat{\beta}RE$ *is* ***consistent and efficient****, while* $\hat{\beta}FE$ is **consistent**
$H_0 : \text{Cov}(c_i, \mathbf{x_{it}}) \neq 0$	$\hat{\beta}RE$ *is* ***inconsistent****, while* $\hat{\beta}FE$ remains **consistent**

Hausman Test

- **Purpose:** Determines whether FE or RE is appropriate.

For the Hausman test to work, you need to assume that

- Strict exogeneity hold
- A4 to hold for u_{it}

Then,

- Hausman test statistic: $H = (\hat{\beta}_{RE} - \hat{\beta}_{FE})'(V(\hat{\beta}_{RE}) - V(\hat{\beta}_{FE}))(\hat{\beta}_{RE} - \hat{\beta}_{FE}) \sim \chi^2_{n(X)}$ where $n(X)$ is the number of parameters for the time-varying regressors.
- **Null hypothesis (H_0):** RE estimator is consistent and efficient.
- **Alternative hypothesis (H_A):** RE estimator is inconsistent, meaning FE should be used.
- **Decision Rule:**
  - If $p < 0.05$: Reject H_0, meaning FE is preferred.
  - If $p \geq 0.05$: Fail to reject H_0, meaning RE can be used.

```r
library(plm)

# Fit FE and RE models
fe_model <- plm(inv ~ value + capital, data = Grunfeld, model = "within")
re_model <- plm(inv ~ value + capital, data = Grunfeld, model = "random")

# Hausman test
phtest(fe_model, re_model)
#>
#>  Hausman Test
#>
#> data:  inv ~ value + capital
#> chisq = 2.3304, df = 2, p-value = 0.3119
#> alternative hypothesis: one model is inconsistent
```

If the null hypothesis is rejected, use FE. If not, RE is appropriate.

11.5.9.3 Summary of Model Assumptions and Consistency

All three estimators (POLS, RE, FE) require:

- A1 Linearity
- A2 Full rank
- A5 Data generation (random sampling) for individuals

However, additional assumptions determine whether the estimator is **consistent and efficient**.

POLS

- **Consistent if:**
  - A3a Exogeneity holds: $E(\mathbf{x}_{it}' u_{it}) = 0$
  - RE assumption holds: $E(\mathbf{x}_{it}' c_i) = 0$
- If A4 Homoskedasticity does not hold: Use cluster-robust standard errors, but POLS is not efficient.

RE

- **Consistent if:**
  - A3a Exogeneity holds: $E(\mathbf{x}_{it}' u_{it}) = 0$
  - RE assumption holds: $E(\mathbf{x}_{it}' c_i) = 0$
- If A4 Homoskedasticity holds: RE is most efficient.
- If A4 Homoskedasticity does not hold: Use cluster-robust standard errors. RE remains more efficient than POLS but is not the most efficient.

FE

- **Consistent if:**
  - A3a Exogeneity holds: $E((\mathbf{x}_{it} - \bar{\mathbf{x}}_{it})'(u_{it} - \bar{u}_{it})) = 0$
- **Limitations:**
  - Cannot estimate the effects of time-constant variables.
  - A4 Homoskedasticity generally does not hold, so cluster-robust SEs are required.

Estimator Selection Guide

Estimator / True Model	POLS	RE	FE
POLS	Consistent	Consistent	Inconsistent
FE	Consistent	Consistent	Consistent
RE	Consistent	Consistent	Inconsistent

11.5.10 Alternative Estimators

Other estimators are available depending on model violations and additional considerations:

- **Violation Estimators**: Adjust for assumption violations.
- **Basic Estimators**: Standard POLS, RE, FE.
- **Instrumental Variable Estimator**: Used for endogeneity.
- **Variable Coefficient Estimator**: Allows varying coefficients.
- **Generalized Method of Moments Estimator**: For dynamic panel models.
- **General Feasible GLS Estimator**: Accounts for heteroskedasticity and serial correlation.
- **Means Groups Estimator**: Averages individual-specific estimates.
- **Common Correlated Effects Mean Group Estimator**: Accounts for cross-sectional dependence.
- **Limited Dependent Variable Estimators**: Used for binary or censored data.

11.5.11 Application

11.5.11.1 `plm` Package

The `plm` package in R is designed for panel data analysis, allowing users to estimate various models, including **pooled OLS**, **fixed effects**, **random effects**, and other specifications commonly used in econometrics.

For a detailed guide, refer to:

- The official vignette on `plm` functions.
- The model components reference.

```r
# Load the package
library("plm")

data("Produc", package = "plm")
```

```r
# Display first few rows
head(Produc)
#>     state year region     pcap     hwy   water    util       pc   gsp    emp
#> 1 ALABAMA 1970      6 15032.67 7325.80 1655.68 6051.20 35793.80 28418 1010.5
#> 2 ALABAMA 1971      6 15501.94 7525.94 1721.02 6254.98 37299.91 29375 1021.9
#> 3 ALABAMA 1972      6 15972.41 7765.42 1764.75 6442.23 38670.30 31303 1072.3
#> 4 ALABAMA 1973      6 16406.26 7907.66 1742.41 6756.19 40084.01 33430 1135.5
#> 5 ALABAMA 1974      6 16762.67 8025.52 1734.85 7002.29 42057.31 33749 1169.8
#> 6 ALABAMA 1975      6 17316.26 8158.23 1752.27 7405.76 43971.71 33604 1155.4
#>   unemp
#> 1   4.7
#> 2   5.2
#> 3   4.7
#> 4   3.9
#> 5   5.5
#> 6   7.7
```

To specify panel data, we define the **individual (cross-sectional) and time identifiers**:

```r
# Convert data to panel format
pdata <- pdata.frame(Produc, index = c("state", "year"))

# Check structure
summary(pdata)
#>          state          year         region         pcap             hwy
#>  ALABAMA    : 17   1970   : 48   5      :136   Min.   :  2627   Min.   : 1827
#>  ARIZONA    : 17   1971   : 48   8      :136   1st Qu.:  7097   1st Qu.: 3858
#>  ARKANSAS   : 17   1972   : 48   4      :119   Median : 17572   Median : 7556
#>  CALIFORNIA : 17   1973   : 48   1      :102   Mean   : 25037   Mean   :10218
#>  COLORADO   : 17   1974   : 48   3      : 85   3rd Qu.: 27692   3rd Qu.:11267
#>  CONNECTICUT: 17   1975   : 48   6      : 68   Max.   :140217   Max.   :47699
#>  (Other)    :714   (Other):528   (Other):170
#>      water              util               pc              gsp
#>  Min.   :  228.5   Min.   :  538.5   Min.   :  4053   Min.   :  4354
#>  1st Qu.:  764.5   1st Qu.: 2488.3   1st Qu.: 21651   1st Qu.: 16503
#>  Median : 2266.5   Median : 7008.8   Median : 40671   Median : 39987
#>  Mean   : 3618.8   Mean   :11199.5   Mean   : 58188   Mean   : 61014
#>  3rd Qu.: 4318.7   3rd Qu.:11598.5   3rd Qu.: 64796   3rd Qu.: 68126
#>  Max.   :24592.3   Max.   :80728.1   Max.   :375342   Max.   :464550
#>
#>       emp              unemp
#>  Min.   :  108.3   Min.   : 2.800
#>  1st Qu.:  475.0   1st Qu.: 5.000
#>  Median : 1164.8   Median : 6.200
```

```
#>  Mean   : 1747.1   Mean   : 6.602
#>  3rd Qu.: 2114.1   3rd Qu.: 7.900
#>  Max.   :11258.0   Max.   :18.000
#>
```

The `plm` package allows for the estimation of several different panel data models.

1. **Pooled OLS Estimator**

A simple **pooled OLS** model assumes a common intercept and ignores individual-specific effects.

```
pooling <- plm(log(gsp) ~ log(pcap) + log(emp) + unemp,
               data = pdata,
               model = "pooling")
summary(pooling)
#> Pooling Model
#>
#> Call:
#> plm(formula = log(gsp) ~ log(pcap) + log(emp) + unemp, data = pdata,
#>     model = "pooling")
#>
#> Balanced Panel: n = 48, T = 17, N = 816
#>
#> Residuals:
#>      Min.   1st Qu.    Median   3rd Qu.      Max.
#> -0.302260 -0.085204 -0.018166  0.051783  0.500144
#>
#> Coefficients:
#>               Estimate Std. Error t-value Pr(>|t|)
#> (Intercept)  2.2124123  0.0790988 27.9703  < 2e-16 ***
#> log(pcap)    0.4121307  0.0216314 19.0525  < 2e-16 ***
#> log(emp)     0.6205834  0.0199495 31.1078  < 2e-16 ***
#> unemp       -0.0035444  0.0020539 -1.7257  0.08478 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Total Sum of Squares:    849.81
#> Residual Sum of Squares: 13.326
#> R-Squared:      0.98432
#> Adj. R-Squared: 0.98426
#> F-statistic: 16990.2 on 3 and 812 DF, p-value: < 2.22e-16
```

2. **Between Estimator**

This estimator takes the average over time for each entity, reducing within-group variation.

```r
between <- plm(log(gsp) ~ log(pcap) + log(emp) + unemp,
               data = pdata, model = "between")
summary(between)
#> Oneway (individual) effect Between Model
#>
#> Call:
#> plm(formula = log(gsp) ~ log(pcap) + log(emp) + unemp, data = pdata,
#>     model = "between")
#>
#> Balanced Panel: n = 48, T = 17, N = 816
#> Observations used in estimation: 48
#>
#> Residuals:
#>      Min.   1st Qu.    Median   3rd Qu.      Max.
#> -0.172055 -0.086456 -0.013203  0.038100  0.394336
#>
#> Coefficients:
#>               Estimate Std. Error t-value  Pr(>|t|)
#> (Intercept)  2.0784403  0.3277756  6.3410 1.063e-07 ***
#> log(pcap)    0.4585009  0.0892620  5.1366 6.134e-06 ***
#> log(emp)     0.5751005  0.0828921  6.9379 1.410e-08 ***
#> unemp       -0.0031585  0.0145683 -0.2168    0.8294
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Total Sum of Squares:    48.875
#> Residual Sum of Squares: 0.65861
#> R-Squared:      0.98652
#> Adj. R-Squared: 0.98561
#> F-statistic: 1073.73 on 3 and 44 DF, p-value: < 2.22e-16
```

3. **First-Differences Estimator**

The **first-differences** model eliminates time-invariant effects by differencing adjacent periods.

```r
firstdiff <- plm(log(gsp) ~ log(pcap) + log(emp) + unemp,
                 data = pdata, model = "fd")
summary(firstdiff)
#> Oneway (individual) effect First-Difference Model
#>
#> Call:
#> plm(formula = log(gsp) ~ log(pcap) + log(emp) + unemp, data = pdata,
#>     model = "fd")
#>
#> Balanced Panel: n = 48, T = 17, N = 816
```

```
#> Observations used in estimation: 768
#>
#> Residuals:
#>       Min.    1st Qu.     Median    3rd Qu.       Max.
#> -0.0846921 -0.0108511  0.0016861  0.0124968  0.1018911
#>
#> Coefficients:
#>               Estimate Std. Error t-value  Pr(>|t|)
#> (Intercept)  0.0101353  0.0013206  7.6749 5.058e-14 ***
#> log(pcap)   -0.0167634  0.0453958 -0.3693     0.712
#> log(emp)     0.8212694  0.0362737 22.6409 < 2.2e-16 ***
#> unemp       -0.0061615  0.0007516 -8.1978 1.032e-15 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Total Sum of Squares:    1.0802
#> Residual Sum of Squares: 0.33394
#> R-Squared:      0.69086
#> Adj. R-Squared: 0.68965
#> F-statistic: 569.123 on 3 and 764 DF, p-value: < 2.22e-16
```

4. **Fixed Effects (Within) Estimator**

Controls for time-invariant heterogeneity by demeaning data within individuals.

```
fixed <- plm(log(gsp) ~ log(pcap) + log(emp) + unemp,
             data = pdata, model = "within")
summary(fixed)
#> Oneway (individual) effect Within Model
#>
#> Call:
#> plm(formula = log(gsp) ~ log(pcap) + log(emp) + unemp, data = pdata,
#>     model = "within")
#>
#> Balanced Panel: n = 48, T = 17, N = 816
#>
#> Residuals:
#>       Min.    1st Qu.     Median    3rd Qu.       Max.
#> -0.1253873 -0.0248746 -0.0054276  0.0184698  0.2026394
#>
#> Coefficients:
#>              Estimate  Std. Error t-value Pr(>|t|)
#> log(pcap)  0.03488447  0.03092191  1.1281   0.2596
#> log(emp)   1.03017988  0.02161353 47.6636   <2e-16 ***
#> unemp     -0.00021084  0.00096121 -0.2194   0.8264
#> ---
```

```
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Total Sum of Squares:    18.941
#> Residual Sum of Squares: 1.3077
#> R-Squared:      0.93096
#> Adj. R-Squared: 0.92645
#> F-statistic: 3438.48 on 3 and 765 DF, p-value: < 2.22e-16
```

5. **Random Effects Estimator**

Accounts for unobserved heterogeneity by modeling it as a random component.

```
random <- plm(log(gsp) ~ log(pcap) + log(emp) + unemp,
              data = pdata, model = "random")
summary(random)
#> Oneway (individual) effect Random Effect Model
#>    (Swamy-Arora's transformation)
#>
#> Call:
#> plm(formula = log(gsp) ~ log(pcap) + log(emp) + unemp, data = pdata,
#>     model = "random")
#>
#> Balanced Panel: n = 48, T = 17, N = 816
#>
#> Effects:
#>                    var  std.dev share
#> idiosyncratic 0.001709 0.041345 0.103
#> individual    0.014868 0.121934 0.897
#> theta: 0.918
#>
#> Residuals:
#>       Min.    1st Qu.     Median    3rd Qu.       Max.
#> -0.1246674 -0.0268273 -0.0049657  0.0214145  0.2389889
#>
#> Coefficients:
#>               Estimate Std. Error z-value Pr(>|z|)
#> (Intercept) 3.10569727 0.14715985 21.1042   <2e-16 ***
#> log(pcap)   0.03708054 0.02747015  1.3498   0.1771
#> log(emp)    1.00937552 0.02103951 47.9752   <2e-16 ***
#> unemp       0.00004806 0.00092301  0.0521   0.9585
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Total Sum of Squares:    24.523
#> Residual Sum of Squares: 1.4425
#> R-Squared:      0.94118
```

```
#> Adj. R-Squared: 0.94096
#> Chisq: 12992.5 on 3 DF, p-value: < 2.22e-16
```

Model Selection and Diagnostic Tests

1. **Lagrange Multiplier Test for Random Effects**

The **Breusch-Pagan LM test** compares random effects with pooled OLS.

- **Null Hypothesis:** OLS is preferred.
- **Alternative Hypothesis:** Random effects model is appropriate.

```
plmtest(pooling, effect = "individual", type = "bp")
#>
#>  Lagrange Multiplier Test - (Breusch-Pagan)
#>
#> data:  log(gsp) ~ log(pcap) + log(emp) + unemp
#> chisq = 4567.1, df = 1, p-value < 2.2e-16
#> alternative hypothesis: significant effects
```

Other test types: `"honda"`, `"kw"`, `"ghm"`. Other effects: `"time"`, `"twoways"`.

2. **Cross-Sectional Dependence Tests**

Breusch-Pagan LM test for cross-sectional dependence

```
pcdtest(fixed, test = "lm")
#>
#>  Breusch-Pagan LM test for cross-sectional dependence in panels
#>
#> data:  log(gsp) ~ log(pcap) + log(emp) + unemp
#> chisq = 6490.4, df = 1128, p-value < 2.2e-16
#> alternative hypothesis: cross-sectional dependence
```

Pesaran's CD statistic

```
pcdtest(fixed, test = "cd")
#>
#>  Pesaran CD test for cross-sectional dependence in panels
#>
#> data:  log(gsp) ~ log(pcap) + log(emp) + unemp
#> z = 37.13, p-value < 2.2e-16
#> alternative hypothesis: cross-sectional dependence
```

3. **Serial Correlation Test (Panel Version of the Breusch-Godfrey Test)**

Used to check for autocorrelation in panel data.

```r
pbgtest(fixed)
#>
#>  Breusch-Godfrey/Wooldridge test for serial correlation in panel models
#>
#> data:  log(gsp) ~ log(pcap) + log(emp) + unemp
#> chisq = 476.92, df = 17, p-value < 2.2e-16
#> alternative hypothesis: serial correlation in idiosyncratic errors
```

4. **Stationarity Test (Augmented Dickey-Fuller Test)**

Checks whether a time series variable is stationary.

```r
library(tseries)
adf.test(pdata$gsp, k = 2)
#>
#>  Augmented Dickey-Fuller Test
#>
#> data:  pdata$gsp
#> Dickey-Fuller = -5.9028, Lag order = 2, p-value = 0.01
#> alternative hypothesis: stationary
```

5. **F-Test for Fixed Effects vs. Pooled OLS**

- **Null Hypothesis:** Pooled OLS is appropriate.
- **Alternative Hypothesis:** Fixed effects model is preferred.

```r
pFtest(fixed, pooling)
#>
#>  F test for individual effects
#>
#> data:  log(gsp) ~ log(pcap) + log(emp) + unemp
#> F = 149.58, df1 = 47, df2 = 765, p-value < 2.2e-16
#> alternative hypothesis: significant effects
```

6. **Hausman Test for Fixed vs. Random Effects**

- **Null Hypothesis:** Random effects are appropriate.
- **Alternative Hypothesis:** Fixed effects are preferred (RE assumptions are violated).

```r
phtest(random, fixed)
#>
#>  Hausman Test
#>
#> data:  log(gsp) ~ log(pcap) + log(emp) + unemp
#> chisq = 84.924, df = 3, p-value < 2.2e-16
#> alternative hypothesis: one model is inconsistent
```

Heteroskedasticity and Robust Standard Errors

1. **Breusch-Pagan Test for Heteroskedasticity**

Tests whether heteroskedasticity is present in the panel dataset.

```
library(lmtest)
bptest(log(gsp) ~ log(pcap) + log(emp) + unemp,  data = pdata)
#>
#>  studentized Breusch-Pagan test
#>
#> data:  log(gsp) ~ log(pcap) + log(emp) + unemp
#> BP = 98.223, df = 3, p-value < 2.2e-16
```

2. **Correcting for Heteroskedasticity**

If heteroskedasticity is detected, use robust standard errors:

For Random Effects Model

```
# Original coefficients
coeftest(random)
#>
#> t test of coefficients:
#>
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 3.10569727 0.14715985 21.1042   <2e-16 ***
#> log(pcap)   0.03708054 0.02747015  1.3498   0.1774
#> log(emp)    1.00937552 0.02103951 47.9752   <2e-16 ***
#> unemp       0.00004806 0.00092301  0.0521   0.9585
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Heteroskedasticity-consistent standard errors
coeftest(random, vcovHC)
#>
#> t test of coefficients:
#>
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 3.10569727 0.23261788 13.3511   <2e-16 ***
#> log(pcap)   0.03708054 0.06125725  0.6053   0.5451
#> log(emp)    1.00937552 0.06395880 15.7817   <2e-16 ***
#> unemp       0.00004806 0.00215219  0.0223   0.9822
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Different HC types
t(sapply(c("HC0", "HC1", "HC2", "HC3", "HC4"), function(x)
    sqrt(diag(vcovHC(random, type = x)))
```

```
))
#>     (Intercept)  log(pcap)   log(emp)       unemp
#> HC0   0.2326179 0.06125725 0.06395880 0.002152189
#> HC1   0.2331901 0.06140795 0.06411614 0.002157484
#> HC2   0.2334857 0.06161618 0.06439057 0.002160392
#> HC3   0.2343595 0.06197939 0.06482756 0.002168646
#> HC4   0.2342815 0.06235576 0.06537813 0.002168867
```

- **HC0**: Default heteroskedasticity-consistent (White's estimator).
- **HC1, HC2, HC3**: Recommended for small samples.
- **HC4**: Useful for small samples with influential observations.

For Fixed Effects Model

```
# Original coefficients
coeftest(fixed)
#>
#> t test of coefficients:
#>
#>              Estimate  Std. Error t value Pr(>|t|)
#> log(pcap)  0.03488447  0.03092191  1.1281   0.2596
#> log(emp)   1.03017988  0.02161353 47.6636   <2e-16 ***
#> unemp     -0.00021084  0.00096121 -0.2194   0.8264
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Heteroskedasticity-consistent standard errors
coeftest(fixed, vcovHC)
#>
#> t test of coefficients:
#>
#>              Estimate  Std. Error t value Pr(>|t|)
#> log(pcap)  0.03488447  0.06661083  0.5237   0.6006
#> log(emp)   1.03017988  0.06413365 16.0630   <2e-16 ***
#> unemp     -0.00021084  0.00217453 -0.0970   0.9228
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Arellano method for robust errors
coeftest(fixed, vcovHC(fixed, method = "arellano"))
#>
#> t test of coefficients:
#>
#>              Estimate  Std. Error t value Pr(>|t|)
#> log(pcap)  0.03488447  0.06661083  0.5237   0.6006
```

```
#> log(emp)   1.03017988  0.06413365 16.0630   <2e-16 ***
#> unemp     -0.00021084  0.00217453 -0.0970   0.9228
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Different HC types
t(sapply(c("HC0", "HC1", "HC2", "HC3", "HC4"), function(x)
    sqrt(diag(vcovHC(fixed, type = x)))
))
#>      log(pcap)   log(emp)       unemp
#> HC0 0.06661083 0.06413365 0.002174525
#> HC1 0.06673362 0.06425187 0.002178534
#> HC2 0.06689078 0.06441024 0.002182114
#> HC3 0.06717278 0.06468886 0.002189747
#> HC4 0.06742431 0.06496436 0.002193150
```

Table 11.17: Summary of Model Selection

Test	Null Hypothesis (H)	Decision Rule
LM Test	OLS is appropriate	Reject H → Use RE
Hausman Test	Random effects preferred	Reject H → Use FE
pFtest	OLS is appropriate	Reject H → Use FE
Breusch-Pagan	No heteroskedasticity	Reject H → Use robust SE

Variance Components Structure

Beyond the standard random effects model, the `plm` package provides additional methods for estimating **variance components models** and instrumental variable techniques for dealing with endogeneity in panel data.

Different estimators for the **variance components structure** exist in the literature, and `plm` allows users to specify them through the `random.method` argument.

Random Effects Estimators:

- `"swar"` (*default*): Swamy and Arora estimator (Swamy and Arora, 1972).
- `"walhus"`: Wallace and Hussain estimator (Wallace and Hussain, 1969).
- `"amemiya"`: Amemiya estimator (Amemiya, 1971).
- `"nerlove"`: Nerlove estimator (Nerlove, 1971) (**Note:** Not available for two-way random effects).

Effects in Panel Models:

- **Individual effects** (*default*).
- **Time effects** (`effect = "time"`).
- **Two-way effects** (`effect = "twoways"`).

```
amemiya <- plm(
    log(gsp) ~ log(pcap) + log(emp) + unemp,
    data = pdata,
    model = "random",
    random.method = "amemiya",
    effect = "twoways"
)

summary(amemiya)
#> Twoways effects Random Effect Model
#>    (Amemiya's transformation)
#>
#> Call:
#> plm(formula = log(gsp) ~ log(pcap) + log(emp) + unemp, data = pdata,
#>     effect = "twoways", model = "random", random.method = "amemiya")
#>
#> Balanced Panel: n = 48, T = 17, N = 816
#>
#> Effects:
#>                    var  std.dev share
#> idiosyncratic 0.001228 0.035039 0.028
#> individual    0.041201 0.202981 0.941
#> time          0.001359 0.036859 0.031
#> theta: 0.9582 (id) 0.8641 (time) 0.8622 (total)
#>
#> Residuals:
#>        Min.     1st Qu.      Median     3rd Qu.        Max.
#> -0.13796209 -0.01951506 -0.00053384  0.01807398  0.20452581
#>
#> Coefficients:
#>               Estimate Std. Error z-value  Pr(>|z|)
#> (Intercept)  3.9581876  0.1767036 22.4001 < 2.2e-16 ***
#> log(pcap)    0.0378443  0.0253963  1.4902  0.136184
#> log(emp)     0.8891887  0.0227677 39.0548 < 2.2e-16 ***
#> unemp       -0.0031568  0.0011240 -2.8086  0.004976 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Total Sum of Squares:    5.3265
#> Residual Sum of Squares: 0.98398
#> R-Squared:      0.81527
```

```
#> Adj. R-Squared: 0.81458
#> Chisq: 3583.53 on 3 DF, p-value: < 2.22e-16
```

The `ercomp()` function retrieves estimates of the **variance components** in a random effects model. Below, we extract the variance decomposition using Amemiya's method:

```
ercomp(log(gsp) ~ log(pcap) + log(emp) + unemp,
       data = pdata,
       method = "amemiya",
       effect = "twoways")
#>                     var  std.dev share
#> idiosyncratic 0.001228 0.035039 0.028
#> individual    0.041201 0.202981 0.941
#> time          0.001359 0.036859 0.031
#> theta: 0.9582 (id) 0.8641 (time) 0.8622 (total)
```

This output includes:

- Variance of the **individual effect**.
- Variance of the **time effect** (if applicable).
- Variance of the **idiosyncratic error**.

Checking Panel Data Balance

Panel datasets may be balanced (each individual has observations for all time periods) or unbalanced (some individuals are missing observations). The `punbalancedness()` function measures the degree of balance in the data, with values closer to 1 indicating a balanced panel (Ahrens and Pincus, 1981).

```
punbalancedness(random)
#> gamma    nu
#>     1     1
```

Instrumental Variables in Panel Data

Instrumental variables (IV) are used to address **endogeneity**, which arises when regressors are correlated with the error term. `plm` provides various IV estimation methods through the `inst.method` argument.

Instrumental Variable Estimators

- `"bvk"`: Balestra-Varadharajan-Krishnakumar estimator (*default*) (Balestra and Varadharajan-Krishnakumar, 1987).
- `"baltagi"`: Baltagi estimator (Baltagi, 1981).

- "am": Amemiya-MaCurdy estimator (Amemiya and MaCurdy, 1986).
- "bms": Breusch-Mizon-Schmidt estimator (Breusch et al., 1989).

Other Estimators in Panel Data Models

Beyond standard fixed effects and random effects models, the `plm` package provides additional estimation techniques tailored for heterogeneous coefficients, dynamic panel models, and feasible generalized least squares (FGLS) methods.

Variable Coefficients Model (`pvcm`)

The variable coefficients model (VCM) allows coefficients to vary across cross-sectional units, accounting for unobserved heterogeneity more flexibly.

Two Estimation Approaches:

- **Fixed effects (`within`)**: Assumes coefficients are constant over time but vary across individuals.
- **Random effects (`random`)**: Assumes coefficients are drawn from a random distribution.

```
fixed_pvcm  <-
    pvcm(log(gsp) ~ log(pcap) + log(emp) + unemp,
         data = pdata,
         model = "within")
random_pvcm <-
    pvcm(log(gsp) ~ log(pcap) + log(emp) + unemp,
         data = pdata,
         model = "random")

summary(fixed_pvcm)
#> Oneway (individual) effect No-pooling model
#>
#> Call:
#> pvcm(formula = log(gsp) ~ log(pcap) + log(emp) + unemp, data = pdata,
#>     model = "within")
#>
#> Balanced Panel: n = 48, T = 17, N = 816
#>
#> Residuals:
#>         Min.      1st Qu.       Median      3rd Qu.         Max.
#> -0.075247625 -0.013247956  0.000666934  0.013852996  0.118966807
#>
#> Coefficients:
#>   (Intercept)        log(pcap)           log(emp)          unemp
#>  Min.   :-3.8868   Min.   :-1.11962   Min.   :0.3790   Min.   :-1.597e-02
```

```
#>  1st Qu.: 0.9917   1st Qu.:-0.38475   1st Qu.:0.8197   1st Qu.:-5.319e-03
#>  Median : 2.9848   Median :-0.03147   Median :1.1506   Median : 5.335e-05
#>  Mean   : 2.8079   Mean   :-0.06028   Mean   :1.1656   Mean   : 9.024e-04
#>  3rd Qu.: 4.3553   3rd Qu.: 0.25573   3rd Qu.:1.3779   3rd Qu.: 8.374e-03
#>  Max.   :12.8800   Max.   : 1.16922   Max.   :2.4276   Max.   : 2.507e-02
#>
#> Total Sum of Squares: 15729
#> Residual Sum of Squares: 0.40484
#> Multiple R-Squared: 0.99997
summary(random_pvcm)
#> Oneway (individual) effect Random coefficients model
#>
#> Call:
#> pvcm(formula = log(gsp) ~ log(pcap) + log(emp) + unemp, data = pdata,
#>     model = "random")
#>
#> Balanced Panel: n = 48, T = 17, N = 816
#>
#> Residuals:
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> -0.23364 -0.03401  0.05558  0.09811  0.19349  1.14326
#>
#> Estimated mean of the coefficients:
#>               Estimate  Std. Error z-value  Pr(>|z|)
#> (Intercept)  2.79030044  0.53104167  5.2544 1.485e-07 ***
#> log(pcap)   -0.04195768  0.08621579 -0.4867    0.6265
#> log(emp)     1.14988911  0.07225221 15.9149 < 2.2e-16 ***
#> unemp        0.00031135  0.00163864  0.1900    0.8493
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Estimated variance of the coefficients:
#>             (Intercept) log(pcap)   log(emp)       unemp
#> (Intercept)  11.2648882 -1.335932  0.2035824  0.00827707
#> log(pcap)    -1.3359322  0.287021 -0.1872915 -0.00345298
#> log(emp)      0.2035824 -0.187291  0.2134845  0.00336374
#> unemp         0.0082771 -0.003453  0.0033637  0.00009425
#>
#> Total Sum of Squares: 15729
#> Residual Sum of Squares: 40.789
#> Multiple R-Squared: 0.99741
#> Chisq: 739.334 on 3 DF, p-value: < 2.22e-16
```

Generalized Method of Moments Estimator (`pgmm`)

The **Generalized Method of Moments** estimator is commonly used for **dy-**

namic panel models, especially when:

- There is concern over **endogeneity** in lagged dependent variables.
- Instrumental variables are used for estimation.

```r
library(plm)

# estimates a dynamic labor demand function using one-step GMM,
# applying lagged variables as instruments
z2 <- pgmm(
    log(emp) ~ lag(log(emp), 1) + lag(log(wage), 0:1) +
        lag(log(capital), 0:1) |
        lag(log(emp), 2:99) +
        lag(log(wage), 2:99) + lag(log(capital), 2:99),
    data = EmplUK,
    effect = "twoways",
    model = "onestep",
    transformation = "ld"
)

summary(z2, robust = TRUE)
#> Twoways effects One-step model System GMM
#>
#> Call:
#> pgmm(formula = log(emp) ~ lag(log(emp), 1) + lag(log(wage), 0:1) +
#>     lag(log(capital), 0:1) | lag(log(emp), 2:99) + lag(log(wage),
#>     2:99) + lag(log(capital), 2:99), data = EmplUK, effect = "twoways",
#>     model = "onestep", transformation = "ld")
#>
#> Unbalanced Panel: n = 140, T = 7-9, N = 1031
#>
#> Number of Observations Used: 1642
#> Residuals:
#>       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
#> -0.7530341 -0.0369030  0.0000000  0.0002882  0.0466069  0.6001503
#>
#> Coefficients:
#>                          Estimate Std. Error z-value  Pr(>|z|)
#> lag(log(emp), 1)         0.935605   0.026295 35.5810 < 2.2e-16 ***
#> lag(log(wage), 0:1)0    -0.630976   0.118054 -5.3448 9.050e-08 ***
#> lag(log(wage), 0:1)1     0.482620   0.136887  3.5257 0.0004224 ***
#> lag(log(capital), 0:1)0  0.483930   0.053867  8.9838 < 2.2e-16 ***
#> lag(log(capital), 0:1)1 -0.424393   0.058479 -7.2572 3.952e-13 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
```

```
#> Sargan test: chisq(100) = 118.763 (p-value = 0.097096)
#> Autocorrelation test (1): normal = -4.808434 (p-value = 1.5212e-06)
#> Autocorrelation test (2): normal = -0.2800133 (p-value = 0.77947)
#> Wald test for coefficients: chisq(5) = 11174.82 (p-value = < 2.22e-16)
#> Wald test for time dummies: chisq(7) = 14.71138 (p-value = 0.039882)
```

Explanation of Arguments:

- `log(emp) ~ lag(log(emp), 1) + lag(log(wage), 0:1) + lag(log(capital), 0:1)`
  → Specifies the **dynamic model**, where `log(emp)` depends on its first lag and contemporaneous plus lagged values of `log(wage)` and `log(capital)`.
- `| lag(log(emp), 2:99) + lag(log(wage), 2:99) + lag(log(capital), 2:99)`
  → Instruments for endogenous regressors, using **further lags**.
- `effect = "twoways"`
  → Includes both **individual and time effects**.
- `model = "onestep"`
  → Uses **one-step GMM** (alternative: `"twostep"` for efficiency gain).
- `transformation = "ld"`
  → Uses **lagged differences** as transformation.

Generalized Feasible Generalized Least Squares Models (`pggls`)

The FGLS estimator (`pggls`) is robust against:

- **Intragroup heteroskedasticity**.
- **Serial correlation** (within groups).

However, it assumes **no cross-sectional correlation** and is most suitable when NNN (cross-sectional units) is much larger than TTT (time periods), i.e., **long panels**.

Random Effects FGLS Model:

```
zz <- pggls(
    log(emp) ~ log(wage) + log(capital),
    data = EmplUK,
    model = "pooling"
)

summary(zz)
#> Oneway (individual) effect General FGLS model
#>
#> Call:
```

```
#> pggls(formula = log(emp) ~ log(wage) + log(capital), data = EmplUK,
#>     model = "pooling")
#>
#> Unbalanced Panel: n = 140, T = 7-9, N = 1031
#>
#> Residuals:
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> -1.80696 -0.36552  0.06181  0.03230  0.44279  1.58719
#>
#> Coefficients:
#>               Estimate Std. Error z-value  Pr(>|z|)
#> (Intercept)   2.023480   0.158468 12.7690 < 2.2e-16 ***
#> log(wage)    -0.232329   0.048001 -4.8401 1.298e-06 ***
#> log(capital)  0.610484   0.017434 35.0174 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> Total Sum of Squares: 1853.6
#> Residual Sum of Squares: 402.55
#> Multiple R-squared: 0.78283
```

Fixed Effects FGLS Model:

```
zz <- pggls(
    log(emp) ~ log(wage) + log(capital),
    data = EmplUK,
    model = "within"
)

summary(zz)
#> Oneway (individual) effect Within FGLS model
#>
#> Call:
#> pggls(formula = log(emp) ~ log(wage) + log(capital), data = EmplUK,
#>     model = "within")
#>
#> Unbalanced Panel: n = 140, T = 7-9, N = 1031
#>
#> Residuals:
#>         Min.      1st Qu.       Median      3rd Qu.         Max.
#> -0.508362414 -0.074254395 -0.002442181  0.076139063  0.601442300
#>
#> Coefficients:
#>               Estimate Std. Error z-value  Pr(>|z|)
#> log(wage)    -0.617617   0.030794 -20.056 < 2.2e-16 ***
#> log(capital)  0.561049   0.017185  32.648 < 2.2e-16 ***
#> ---
```

```
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> Total Sum of Squares: 1853.6
#> Residual Sum of Squares: 17.368
#> Multiple R-squared: 0.99063
```

Key Considerations:

- Efficient under the assumption of homoskedasticity.
- Inefficient if there is group-wise heteroskedasticity.
- Ideal for large-N, small-T panels.

Table 11.18: Summary of Alternative Panel Data Estimators

Estimator	Method	Application
Variable Coefficients (`pvcm`)	Fixed (`within`), Random (`random`)	Allows coefficients to vary across individuals.
GMM (`pgmm`)	One-step, Two-step	Used in dynamic models with endogeneity.
Feasible GLS (`pggls`)	Fixed (`within`), Random (`pooling`)	Handles heteroskedasticity and serial correlation but assumes no cross-sectional correlation.

11.5.11.2 `fixest` Package

The `fixest` package provides **efficient** and **flexible** methods for estimating **fixed effects** and generalized linear models in panel data. It is optimized for handling large datasets with high-dimensional fixed effects and allows for multiple model estimation, robust standard errors, and split-sample estimation.

For further details, refer to the official fixest vignette.

Available Estimation Functions in `fixest`

Function	Model Type
`feols`	Fixed effects OLS (linear regression)
`feglm`	Generalized linear models (GLMs)
`femlm`	Maximum likelihood estimation (MLE)
`feNmlm`	Non-linear models (non-linear in RHS parameters)
`fepois`	Poisson fixed-effects regression
`fenegbin`	Negative binomial fixed-effects regression

Note: These functions work only for `fixest` objects.

```r
library(fixest)
data(airquality)

# Setting a variable dictionary for output labeling
setFixest_dict(
    c(
        Ozone   = "Ozone (ppb)",
        Solar.R = "Solar Radiation (Langleys)",
        Wind    = "Wind Speed (mph)",
        Temp    = "Temperature"
    )
)

# Fixed effects OLS with stepwise estimation and clustering
est <- feols(
    Ozone ~ Solar.R + sw0(Wind + Temp) | csw(Month, Day),
    data = airquality,
    cluster = ~ Day
)

# Display results
etable(est)
#>                                          est.1              est.2
#> Dependent Var.:                     Ozone (ppb)        Ozone (ppb)
#>
#> Solar Radiation (Langleys) 0.1148*** (0.0234)   0.0522* (0.0202)
#> Wind Speed (mph)                               -3.109*** (0.7986)
#> Temperature                                     1.875*** (0.3671)
#> Fixed-Effects:             ------------------ ------------------
#> Month                                      Yes                Yes
#> Day                                         No                 No
#> _________________________ __________________ __________________
#> S.E.: Clustered                        by: Day            by: Day
#> Observations                               111                111
#> R2                                     0.31974            0.63686
#> Within R2                              0.12245            0.53154
#>
#>                                          est.3              est.4
#> Dependent Var.:                     Ozone (ppb)        Ozone (ppb)
#>
#> Solar Radiation (Langleys) 0.1078** (0.0329)   0.0509* (0.0236)
#> Wind Speed (mph)                               -3.289*** (0.7777)
#> Temperature                                     2.052*** (0.2415)
#> Fixed-Effects:             ------------------ ------------------
#> Month                                      Yes                Yes
```

```
#> Day                                        Yes                 Yes
#> ------------------------- ----------------- ------------------
#> S.E.: Clustered                      by: Day            by: Day
#> Observations                             111                111
#> R2                                   0.58018            0.81604
#> Within R2                            0.12074            0.61471
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Output in LaTeX format
etable(est, tex = TRUE)
#> \begingroup
#> \centering
#> \begin{tabular}{lcccc}
#>    \tabularnewline \midrule \midrule
#>    Dependent Variable: & \multicolumn{4}{c}{Ozone (ppb)}\\
#>    Model:                     & (1)            & (2)            & (3)            &
#>    \midrule
#>    \emph{Variables}\\
#>    Solar Radiation (Langleys) & 0.1148$^{***}$ & 0.0522$^{**}$  & 0.1078$^{***}$ &
#>                               & (0.0234)       & (0.0202)       & (0.0329)       &
#>    Wind Speed (mph)           &                & -3.109$^{***}$ &                &
#>                               &                & (0.7986)       &                &
#>    Temperature                &                & 1.875$^{***}$  &                &
#>                               &                & (0.3671)       &                &
#>    \midrule
#>    \emph{Fixed-effects}\\
#>    Month                      & Yes            & Yes            & Yes            &
#>    Day                        &                &                & Yes            &
#>    \midrule
#>    \emph{Fit statistics}\\
#>    Observations               & 111            & 111            & 111            &
#>    R$^2$                      & 0.31974        & 0.63686        & 0.58018        &
#>    Within R$^2$               & 0.12245        & 0.53154        & 0.12074        &
#>    \midrule \midrule
#>    \multicolumn{5}{l}{\emph{Clustered (Day) standard-errors in parentheses}}\\
#>    \multicolumn{5}{l}{\emph{Signif. Codes: ***: 0.01, **: 0.05, *: 0.1}}\\
#> \end{tabular}
#> \par\endgroup

# Extract fixed-effects coefficients
fixedEffects <- fixef(est[[1]])
summary(fixedEffects)
#> Fixed_effects coefficients
#> Number of fixed-effects for variable Month is 5.
```

```
#>  Mean = 19.6 Variance = 272
#>
#> COEFFICIENTS:
#>   Month:     5     6     7     8     9
#>          3.219 8.288 34.26 40.12 12.13

# View fixed effects for one dimension
fixedEffects$Month
#>         5         6         7         8         9
#>  3.218876  8.287899 34.260812 40.122257 12.130971

# Plot fixed effects
plot(fixedEffects)
```

This example demonstrates:

- Fixed effects estimation (`| csw(Month, Day)`).
- Stepwise selection (`sw0(Wind + Temp)`).
- Clustering of standard errors (`cluster = ~ Day`).
- Extracting and plotting fixed effects.

Multiple Model Estimation

1. **Estimating Multiple Dependent Variables (LHS)**

Use `feols()` to estimate models with multiple dependent variables simultaneously:

```
etable(feols(c(Sepal.Length, Sepal.Width) ~
                 Petal.Length + Petal.Width,
             data = iris))
#>                 feols(c(Sepal.L..1 feols(c(Sepal.Le..2
#> Dependent Var.:       Sepal.Length         Sepal.Width
#>
#> Constant         4.191*** (0.0970)   3.587*** (0.0937)
#> Petal.Length    0.5418*** (0.0693) -0.2571*** (0.0669)
#> Petal.Width      -0.3196* (0.1605)    0.3640* (0.1550)
#> _______________ __________________ __________________
#> S.E. type                     IID                 IID
#> Observations                  150                 150
#> R2                        0.76626             0.21310
#> Adj. R2                   0.76308             0.20240
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Alternatively, define a list of dependent variables and loop over them:

```
depvars <- c("Sepal.Length", "Sepal.Width")

res <- lapply(depvars, function(var) {
    feols(xpd(..lhs ~ Petal.Length + Petal.Width, ..lhs = var), data = iris)
})

etable(res)
#>                            model 1             model 2
#> Dependent Var.:       Sepal.Length         Sepal.Width
#>
#> Constant         4.191*** (0.0970)   3.587*** (0.0937)
#> Petal.Length    0.5418*** (0.0693) -0.2571*** (0.0669)
#> Petal.Width      -0.3196* (0.1605)    0.3640* (0.1550)
#> _______________ __________________ __________________
#> S.E. type                     IID                 IID
#> Observations                  150                 150
#> R2                        0.76626             0.21310
#> Adj. R2                   0.76308             0.20240
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2. **Estimating Multiple Specifications (RHS)**

Use **stepwise functions** to estimate different model specifications efficiently.

Options to write the functions

- `sw` (stepwise): sequentially analyze each elements
  - `y ~ sw(x1, x2)` will be estimated as `y ~ x1` and `y ~ x2`
- `sw0` (stepwise 0): similar to `sw` but also estimate a model without the elements in the set first
  - `y ~ sw(x1, x2)` will be estimated as `y ~ 1` and `y ~ x1` and `y ~ x2`
- `csw` (cumulative stepwise): sequentially add each element of the set to the formula
  - `y ~ csw(x1, x2)` will be estimated as `y ~ x1` and `y ~ x1 + x2`
- `csw0` (cumulative stepwise 0): similar to `csw` but also estimate a model without the elements in the set first
  - `y ~ csw(x1, x2)` will be estimated as `y~ 1 y ~ x1` and `y ~ x1 + x2`
- `mvsw` (multiverse stepwise): all possible combination of the elements in the set (it will get large very quick).
  - `mvsw(x1, x2, x3)` will be `sw0(x1, x2, x3, x1 + x2, x1 + x3, x2 + x3, x1 + x2 + x3)`

Stepwise Function	Description
`sw(x1, x2)`	Sequentially estimates models with each element separately.
`sw0(x1, x2)`	Same as `sw()`, but also estimates a **baseline model** without the elements.
`csw(x1, x2)`	Sequentially adds each element to the formula.
`csw0(x1, x2)`	Same as `csw()`, but also includes a **baseline model**.
`mvsw(x1, x2, x3)`	Estimates **all possible combinations** of the variables.

```r
# Example: Cumulative Stepwise Estimation
etable(feols(Ozone ~ csw(Solar.R, Wind, Temp), data = airquality))
#>                              feols(Ozone ~ c..1 feols(Ozone ~ c..2
#> Dependent Var.:                     Ozone (ppb)        Ozone (ppb)
#>
#> Constant                         18.60** (6.748)   77.25*** (9.068)
#> Solar Radiation (Langleys) 0.1272*** (0.0328) 0.1004*** (0.0263)
#> Wind Speed (mph)                                -5.402*** (0.6732)
#> Temperature
#> _________________________ __________________ __________________
#> S.E. type                                 IID                IID
#> Observations                              111                111
#> R2                                    0.12134            0.44949
```

```
#> Adj. R2                                0.11328            0.43930
#>
#>                            feols(Ozone ~ c..3
#> Dependent Var.:                   Ozone (ppb)
#>
#> Constant                     -64.34** (23.05)
#> Solar Radiation (Langleys)   0.0598* (0.0232)
#> Wind Speed (mph)           -3.334*** (0.6544)
#> Temperature                 1.652*** (0.2535)
#> _________________________ __________________
#> S.E. type                                 IID
#> Observations                              111
#> R2                                    0.60589
#> Adj. R2                               0.59484
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Split-Sample Estimation

Estimate separate regressions for different subgroups in the dataset using `fsplit`.

```
etable(feols(Ozone ~ Solar.R + Wind, fsplit = ~ Month, data = airquality))
#>                            feols(Ozone ~ S..1 feols(Ozone ..2 feols(Ozone ~..3
#> Sample (Month)                    Full sample               5                6
#> Dependent Var.:                   Ozone (ppb)     Ozone (ppb)      Ozone (ppb)
#>
#> Constant                     77.25*** (9.068)  50.55* (18.30)    8.997 (16.83)
#> Solar Radiation (Langleys) 0.1004*** (0.0263) 0.0294 (0.0379) 0.1518. (0.0676)
#> Wind Speed (mph)           -5.402*** (0.6732) -2.762* (1.300)  -0.6177 (1.674)
#> _________________________ __________________ _______________ ________________
#> S.E. type                                 IID             IID              IID
#> Observations                              111              24                9
#> R2                                    0.44949         0.22543          0.52593
#> Adj. R2                               0.43930         0.15166          0.36790
#>
#>                            feols(Ozone ~ ..4 feols(Ozone ~ ..5
#> Sample (Month)                            7                 8
#> Dependent Var.:                   Ozone (ppb)       Ozone (ppb)
#>
#> Constant                     88.39*** (20.81)  95.76*** (19.83)
#> Solar Radiation (Langleys)  0.1135. (0.0582) 0.2146** (0.0654)
#> Wind Speed (mph)           -6.319*** (1.559) -8.228*** (1.528)
#> _________________________ _________________ _________________
#> S.E. type                                 IID               IID
```

```
#> Observations                                26                23
#> R2                                     0.52423           0.70640
#> Adj. R2                                0.48286           0.67704
#>
#>                               feols(Ozone ~ ..6
#> Sample (Month)                                9
#> Dependent Var.:                     Ozone (ppb)
#>
#> Constant                        67.10*** (14.35)
#> Solar Radiation (Langleys)       0.0373 (0.0463)
#> Wind Speed (mph)               -4.161*** (1.071)
#> _________________________ _________________
#> S.E. type                                    IID
#> Observations                                 29
#> R2                                     0.38792
#> Adj. R2                                0.34084
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This estimates models separately for each **Month** in the dataset.

Robust Standard Errors in `fixest`

`fixest` supports a variety of **robust standard error estimators**, including:

- `iid`: errors are homoskedastic and independent and identically distributed
- `hetero`: errors are heteroskedastic using White correction
- `cluster`: errors are correlated within the cluster groups
- `newey_west`: (Newey and West, 1986) use for time series or panel data. Errors are heteroskedastic and serially correlated.
  - `vcov = newey_west ~ id + period` where `id` is the subject id and `period` is time period of the panel.
  - to specify lag period to consider `vcov = newey_west(2) ~ id + period` where we're considering 2 lag periods.
- `driscoll_kraay` (Driscoll and Kraay, 1998) use for panel data. Errors are cross-sectionally and serially correlated.
  - `vcov = discoll_kraay ~ period`
- `conley`: (Conley, 1999) for cross-section data. Errors are spatially correlated
  - `vcov = conley ~ latitude + longitude`

  - to specify the distance cutoff, `vcov = vcov_conley(lat = "lat", lon = "long", cutoff = 100, distance = "spherical")`, which will use the `conley()` helper function.

- `hc`: from the `sandwich` package
  - `vcov = function(x) sandwich::vcovHC(x, type = "HC1")`

To let R know which SE estimation you want to use, insert `vcov = vcov_type ~ variables`

Example: Newey-West Standard Errors

```
etable(feols(
    Ozone ~ Solar.R + Wind,
    data = airquality,
    vcov = newey_west ~ Month + Day
))
#>                              feols(Ozone ~ So..
#> Dependent Var.:                     Ozone (ppb)
#>
#> Constant                       77.25*** (10.03)
#> Solar Radiation (Langleys) 0.1004*** (0.0258)
#> Wind Speed (mph)           -5.402*** (0.8353)
#> _________________________ __________________
#> S.E. type                    Newey-West (L=2)
#> Observations                              111
#> R2                                    0.44949
#> Adj. R2                               0.43930
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Specify the number of lag periods to consider:

```
etable(feols(
    Ozone ~ Solar.R + Wind,
    data = airquality,
    vcov = newey_west(2) ~ Month + Day
))
#>                              feols(Ozone ~ So..
#> Dependent Var.:                     Ozone (ppb)
#>
#> Constant                       77.25*** (10.03)
#> Solar Radiation (Langleys) 0.1004*** (0.0258)
#> Wind Speed (mph)           -5.402*** (0.8353)
#> _________________________ __________________
#> S.E. type                    Newey-West (L=2)
#> Observations                              111
#> R2                                    0.44949
```

```
#> Adj. R2                                0.43930
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Conley Spatial Correlation: `vcov = conley ~ latitude + longitude`

To specify a distance cutoff: `vcov = vcov_conley(lat = "lat", lon = "long", cutoff = 100, distance = "spherical")`

Using Standard Errors from the `sandwich` Package

```
etable(feols(
    Ozone ~ Solar.R + Wind,
    data = airquality,
    vcov = function(x)
        sandwich::vcovHC(x, type = "HC1")
))
#>                              feols(Ozone ~ So..
#> Dependent Var.:                     Ozone (ppb)
#>
#> Constant                       77.25*** (9.590)
#> Solar Radiation (Langleys) 0.1004*** (0.0231)
#> Wind Speed (mph)           -5.402*** (0.8134)
#> _________________________ ___________________
#> S.E. type                  vcovHC(type="HC1")
#> Observations                               111
#> R2                                     0.44949
#> Adj. R2                                0.43930
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Small Sample Correction

Apply small sample adjustments using `ssc()`:

```
etable(feols(
    Ozone ~ Solar.R + Wind,
    data = airquality,
    ssc = ssc(adj = TRUE, cluster.adj = TRUE)
))
#>                              feols(Ozone ~ So..
#> Dependent Var.:                     Ozone (ppb)
#>
#> Constant                       77.25*** (9.068)
#> Solar Radiation (Langleys) 0.1004*** (0.0263)
#> Wind Speed (mph)           -5.402*** (0.6732)
#> _________________________ ___________________
#> S.E. type                                  IID
```

```
#> Observations                              111
#> R2                                    0.44949
#> Adj. R2                               0.43930
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This corrects for **bias** when working with small samples.

11.6 Choosing the Right Type of Data

Selecting the appropriate data type depends on:

1. **Research Questions**: Do you need to understand changes over time at the individual level (panel) or just a snapshot comparison at one point (cross-sectional)?
2. **Resources**: Longitudinal or panel studies can be resource-intensive.
3. **Time Constraints**: If you need fast results, cross-sectional or repeated cross-sectional might be more practical.
4. **Analytical Goals**: Time-series forecasting, causal inference, or descriptive comparison each has different data requirements.
5. **Availability**: Sometimes only secondary or repeated cross-sectional data is available, which constrains the design.

11.7 Data Quality and Ethical Considerations

Regardless of data type, **data quality** is crucial. Poor data—be it incomplete, biased, or improperly measured—can lead to incorrect conclusions. Researchers should:

- **Ensure Validity and Reliability**: Use well-designed instruments and consistent measurement techniques.
- **Address Missing Data**: Apply appropriate imputation methods if feasible.
- **Manage Attrition** (in Panel Data): Consider weighting or sensitivity analyses to deal with dropouts.
- **Check Representativeness**: Especially in cross-sectional and repeated cross-sectional surveys, ensure sampling frames match the target population.
- **Protect Confidentiality and Privacy**: Particularly in panel studies with repeated contact, store data securely and follow ethical guidelines.

- **Obtain Proper Consent**: Inform participants about study details, usage of data, and rights to withdraw.

Chapter 12

Variable Transformation

12.1 Continuous Variables

Transforming continuous variables can be useful for various reasons, including:

- **Changing the scale of variables** to make them more interpretable or comparable.
- **Reducing skewness** to approximate a normal distribution, which can improve statistical inference.
- **Stabilizing variance** in cases of heteroskedasticity.
- **Enhancing interpretability** in business applications (e.g., logarithmic transformations for financial data).

12.1.1 Standardization (Z-score Normalization)

A common transformation to center and scale data:

$$x_i' = \frac{x_i - \bar{x}}{s}$$

where:

- x_i is the original value,
- $\bar{x}$ is the sample mean,
- s is the sample standard deviation.

When to Use:

- When variables have different units of measurement and need to be on a common scale.

- When a few large numbers dominate the dataset.

12.1.2 Min-Max Scaling (Normalization)

Rescales data to a fixed range, typically $[0, 1]$:

$$x_i' = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}$$

When to Use:

- When working with fixed-interval data (e.g., percentages, proportions).
- When preserving relative relationships between values is important.
- Caution: This method is sensitive to outliers, as extreme values determine the range.

12.1.3 Square Root and Cube Root Transformations

Useful for handling **positive skewness** and **heteroskedasticity**:

- **Square root:** Reduces moderate skewness and variance.
- **Cube root:** Works on more extreme skewness and allows negative values.

Common Use Cases:

- **Frequency count data** (e.g., website visits, sales transactions).
- **Data with many small values or zeros** (e.g., income distributions in microfinance).

12.1.4 Logarithmic Transformation

Logarithmic transformations are particularly useful for handling highly skewed data. They compress large values while expanding small values, which helps with heteroskedasticity and normality assumptions.

12.1.4.1 Common Log Transformations

Formula	When to Use
$x_i' = \log(x_i)$	When all values are positive.
$x_i' = \log(x_i + 1)$	When data contains zeros.
$x_i' = \log(x_i + c)$	Choosing c depends on context.

Formula	When to Use
$x_i' = \frac{x_i}{\lvert x_i \rvert} \log \lvert x_i \rvert$	When data contains negative values.
$x_i'^{\lambda} = \log(x_i + \sqrt{x_i^2 + \lambda})$	Generalized log transformation.

Selecting the constant c is critical:

- If c is too large, it can obscure the true nature of the data.
- If c is too small, the transformation might not effectively reduce skewness.

From a statistical modeling perspective:

- **For inference-based models**, the choice of c can significantly impact the fit. See (Ekwaru and Veugelers, 2018).
- **In causal inference** (e.g., DID, IV), improper log transformations (e.g., logging zero values) can introduce bias (Chen and Roth, 2023).

12.1.4.2 When is Log Transformation Problematic?

- When **zero values have a meaningful interpretation** (e.g., income of unemployed individuals).
- When **data are censored** (e.g., income data truncated at reporting thresholds).
- When **measurement error exists** (e.g., rounding errors from survey responses).

If zeros are **small but meaningful** (e.g., revenue from startups), then using $\log(x + c)$ may be acceptable.

```r
library(tidyverse)

# Load dataset
data(cars)

# Original speed values
head(cars$speed)
#> [1] 4 4 7 7 8 9

# Log transformation (basic)
log(cars$speed) %>% head()
#> [1] 1.386294 1.386294 1.945910 1.945910 2.079442 2.197225

# Log transformation for zero-inflated data
log1p(cars$speed) %>% head()
#> [1] 1.609438 1.609438 2.079442 2.079442 2.197225 2.302585
```

12.1.5 Exponential Transformation

The exponential transformation is useful when data exhibit negative skewness or when an underlying logarithmic trend is suspected, such as in survival analysis and decay models.

When to Use:

- Negatively skewed distributions.
- Processes that follow an exponential trend (e.g., population growth, depreciation of assets).

12.1.6 Power Transformation

Power transformations help adjust skewness, particularly for negatively skewed data.

When to Use:

- When variables have a negatively skewed distribution.
- When the relationship between variables is non-linear.

Common power transformations include:

- Square transformation: x^2 (moderate adjustment).
- Cubic transformation: x^3 (stronger adjustment).
- Fourth-root transformation: $x^{1/4}$ (more subtle than square root).

12.1.7 Inverse (Reciprocal) Transformation

The inverse transformation is useful for handling platykurtic (flat) distributions or positively skewed data.

Formula:

$$x_i' = \frac{1}{x_i}$$

When to Use:

- Reducing extreme values in positively skewed distributions.
- Ratio data (e.g., speed = distance/time).
- When the variable has a natural lower bound (e.g., time to completion).

```r
data(cars)

# Original distribution
head(cars$dist)
#> [1]  2 10  4 22 16 10
plot(cars$dist)
```

```r
# Reciprocal transformation
plot(1 / cars$dist)
```

12.1.8 Hyperbolic Arcsine Transformation

The arcsinh (inverse hyperbolic sine) transformation is useful for handling proportion variables (0-1) and skewed distributions. It behaves similarly to the logarithmic transformation but has the advantage of handling zero and negative values.

Formula:

$$\text{arcsinh}(Y) = \log(\sqrt{1 + Y^2} + Y)$$

When to Use:

- Proportion variables (e.g., market share, probability estimates).
- Data with extreme skewness where log transformation is problematic.
- Variables containing zeros or negative values (unlike log, arcsinh handles zeros naturally).
- Alternative to log transformation for handling zeros.

```
# Visualize original distribution
cars$dist %>% hist()
```

```
# Alternative histogram
cars$dist %>% MASS::truehist()
```

```
# Apply arcsinh transformation
as_dist <- bestNormalize::arcsinh_x(cars$dist)
as_dist
#> Standardized asinh(x) Transformation with 50 nonmissing obs.:
#>  Relevant statistics:
```

```
#>  - mean (before standardization) = 4.230843
#>  - sd (before standardization) = 0.7710887
as_dist$x.t %>% hist()
```

Paper	Interpretation
Azoulay et al. (2019)	Elasticity
Faber and Gaubert (2019)	Percentage
Hjort and Poulsen (2019)	Percentage
Johnson (2020)	Percentage
Beerli et al. (2021)	Percentage
Norris et al. (2021)	Percentage
Berkouwer and Dean (2022)	Percentage
Cabral et al. (2022)	Elasticity
Carranza et al. (2022)	Percentage
Mirenda et al. (2022)	Percentage

Consider a simple regression model:

$$Y = \beta X + \epsilon$$

When both Y and X are transformed:

- The coefficient estimate β represents **elasticity**: *A 1% increase in* X leads to a β% change in Y.

When only Y is transformed:

- The coefficient estimate represents a **percentage change in** Y for a one-unit change in X.

This makes the arcsinh transformation particularly valuable for **log-linear models** where zero values exist.

12.1.9 Ordered Quantile Normalization (Rank-Based Transformation)

The Ordered Quantile Normalization (OQN) technique transforms data into a normal distribution using rank-based methods (Bartlett, 1947).

Formula:

$$x_i' = \Phi^{-1}\left(\frac{\text{rank}(x_i) - 1/2}{\text{length}(x)}\right)$$

where Φ^{-1} is the inverse normal cumulative distribution function.

When to Use:

- When data are **heavily skewed** or **contain extreme values**.
- When normality is required for **parametric tests**.

```
ord_dist <- bestNormalize::orderNorm(cars$dist)
ord_dist
#> orderNorm Transformation with 50 nonmissing obs and ties
#>  - 35 unique values
#>  - Original quantiles:
#>   0%  25%  50%  75% 100%
#>    2   26   36   56  120
ord_dist$x.t %>% hist()
```

12.1.10 Lambert W x F Transformation

The Lambert W transformation is a more advanced method that normalizes data by removing skewness and heavy tails.

When to Use:

- When traditional transformations (e.g., log, Box-Cox) fail.
- When dealing with heavy-tailed distributions.

```
data(cars)
head(cars$dist)
#> [1]  2 10  4 22 16 10
cars$dist %>% hist()
```

```
# Apply Lambert W transformation
l_dist <- LambertW::Gaussianize(cars$dist)
l_dist %>% hist()
```

12.1.11 Inverse Hyperbolic Sine Transformation

The Inverse Hyperbolic Sine (IHS) transformation is similar to the log transformation but handles zero and negative values (Johnson, 1949).

Formula:

$$f(x, \theta) = \frac{\sinh^{-1}(\theta x)}{\theta} = \frac{\log(\theta x + (\theta^2 x^2 + 1)^{1/2})}{\theta}$$

When to Use:

- When data contain zeros or negative values.
- Alternative to log transformation in economic and financial modeling.

12.1.12 Box-Cox Transformation

The Box-Cox transformation is a power transformation designed to improve linearity and normality (Manly, 1976; Bickel and Doksum, 1981; Box and Cox, 1981).

Formula:

$$x_i'^{\lambda} = \begin{cases} \frac{x_i^{\lambda} - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(x_i) & \text{if } \lambda = 0 \end{cases}$$

When to Use:

- To fix non-linearity in the error terms of regression models.
- When data are strictly positive

```r
library(MASS)
data(cars)
mod <- lm(cars$speed ~ cars$dist, data = cars)

# Check residuals
plot(mod)
```

Residuals vs Fitted
Residuals
Fitted values
lm(cars$speed ~ cars$dist)
39
2
23
5
0
−5
10
15
20
25

Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(cars$speed ~ cars$dist)
39
2
23
2
1
0
−1
−2

```r
# Find optimal lambda
bc <- boxcox(mod, lambda = seq(-3, 3))
```

```
best_lambda <- bc$x[which.max(bc$y)]

# Apply transformation
mod_lambda = lm(cars$speed ^ best_lambda ~ cars$dist, data = cars)
plot(mod_lambda)
```

Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(cars$speed^best_lambda ~ cars$dist)
23
45
39

Scale−Location
√|Standardized residuals|
Fitted values
lm(cars$speed^best_lambda ~ cars$dist)
23
39
45

For the **two-parameter Box-Cox transformation**, we use:

$$x_i'(\lambda_1, \lambda_2) = \begin{cases} \frac{(x_i+\lambda_2)^{\lambda_1}-1}{\lambda_1} & \text{if } \lambda_1 \neq 0 \\ \log(x_i + \lambda_2) & \text{if } \lambda_1 = 0 \end{cases}$$

```
# Two-parameter Box-Cox transformation
two_bc <- geoR::boxcoxfit(cars$speed)
two_bc
#> Fitted parameters:
#>    lambda      beta   sigmasq
#>  1.028798 15.253008 31.935297
#>
#> Convergence code returned by optim: 0
plot(two_bc)
```

12.1.13 Yeo-Johnson Transformation

Similar to Box-Cox (when $\lambda = 1$), but allows for negative values.

Formula:

$$x_i'^{\lambda} = \begin{cases} \frac{(x_i+1)^{\lambda}-1}{\lambda} & \text{if } \lambda \neq 0, x_i \geq 0 \\ \log(x_i+1) & \text{if } \lambda = 0, x_i \geq 0 \\ \frac{-[(-x_i+1)^{2-\lambda}-1]}{2-\lambda} & \text{if } \lambda \neq 2, x_i < 0 \\ -\log(-x_i+1) & \text{if } \lambda = 2, x_i < 0 \end{cases}$$

```
data(cars)
yj_speed <- bestNormalize::yeojohnson(cars$speed)
yj_speed$x.t %>% hist()
```

.

12.1.14 RankGauss Transformation

A rank-based transformation that maps values to a normal distribution.

When to Use:

- To handle skewed data while preserving rank order.

12.1.15 Automatically Choosing the Best Transformation

The `bestNormalize` package selects the **best transformation** for a given dataset.

```
bestdist <- bestNormalize::bestNormalize(cars$dist)
bestdist$x.t %>% hist()
```

12.2 Categorical Variables

Transforming categorical variables into numerical representations is essential for machine learning models and statistical analysis. The key objectives include:

- Converting categorical data into a format suitable for numerical models.
- Improving model interpretability and performance.
- Handling high-cardinality categorical variables efficiently.

There are multiple ways to transform categorical variables, each with its advantages and use cases. The choice depends on factors like **cardinality**, **ordinality**, and **model type**.

12.2.1 One-Hot Encoding (Dummy Variables)

Creates binary indicator variables for each category.

Formula:
For a categorical variable with k unique values, create k binary columns:

$$x_i' = \begin{cases} 1 & \text{if } x_i = \text{category} \\ 0 & \text{otherwise} \end{cases}$$

When to Use:

- Low-cardinality categorical variables (e.g., “Red”, “Blue”, “Green”).

- Tree-based models (e.g., Random Forest, XGBoost).
- Linear regression models (dummy variables prevent information loss).

```
library(caret)

data(iris)
dummy_vars <- dummyVars(~ Species, data = iris)
one_hot_encoded <- predict(dummy_vars, newdata = iris)
head(one_hot_encoded)
#>   Species.setosa Species.versicolor Species.virginica
#> 1              1                  0                 0
#> 2              1                  0                 0
#> 3              1                  0                 0
#> 4              1                  0                 0
#> 5              1                  0                 0
#> 6              1                  0                 0
```

12.2.2 Label Encoding

Assigns integer values to categories.

Formula:
If a categorical variable has k unique values:

$$\text{Category} \rightarrow \text{Integer}$$

Example:

Category	Encoded Value
Red	1
Blue	2
Green	3

When to Use:

- Ordinal categorical variables (e.g., "Low", "Medium", "High").
- Neural networks (use embeddings instead of one-hot).
- Memory-efficient encoding for high-cardinality features.

```
iris$Species_encoded <- as.numeric(factor(iris$Species))
head(iris$Species_encoded)
#> [1] 1 1 1 1 1 1
```

12.2.3 Feature Hashing (Hash Encoding)

Maps categories to a **fixed number of hash bins**, reducing memory usage.

When to Use:

- High-cardinality categorical variables (e.g., user IDs, URLs).
- Scenarios where an exact category match isn't needed.
- Sparse models (e.g., text data in NLP).

```r
library(text2vec)
library(Matrix)

data(iris)

# Convert the 'Species' factor to character tokens
tokens <- word_tokenizer(as.character(iris$Species))

# Create an iterator over tokens
it <- itoken(tokens, progressbar = FALSE)

# Define the hash_vectorizer with a specified hash size (8 in this case)
vectorizer <- hash_vectorizer(hash_size = 8)

# Create a Document-Term Matrix (DTM) using the hashed features
hashed_dtm <- create_dtm(it, vectorizer)

# Inspect the first few rows of the hashed feature matrix
head(hashed_dtm)
#> 6 x 8 sparse Matrix of class "dgCMatrix"
#>
#> 1 . . . . . . 1 .
#> 2 . . . . . . 1 .
#> 3 . . . . . . 1 .
#> 4 . . . . . . 1 .
#> 5 . . . . . . 1 .
#> 6 . . . . . . 1 .
```

- **word_tokenizer**: This function splits the character vector into tokens. Since `iris$Species` is already a categorical variable with values like `"setosa"`, `"versicolor"`, and `"virginica"`, each value becomes a token.
- **itoken**: Creates an iterator over the tokens.
- **hash_vectorizer**: Sets up a hashing vectorizer that transforms tokens into a sparse feature space of size `2^3 = 8` (because `hash_size = 8` means 2^8 bins; if you intend exactly 8 bins, you might adjust the parameter accordingly).

- **create_dtm**: Builds the document-term matrix (which in this case is analogous to a feature matrix for each observation).

12.2.4 Binary Encoding

Converts categories to binary representations and distributes them across multiple columns.

Example:
For four categories ("A", "B", "C", "D"):

Category	Binary Code	Encoded Columns
A	00	0, 0
B	01	0, 1
C	10	1, 0
D	11	1, 1

When to Use:

- High-cardinality categorical features (less memory than one-hot encoding).
- Tree-based models (preserves some ordinal information).

```
library(mltools)
library(data.table)

# Convert the Species column to a data.table and perform one-hot encoding
binary_encoded <- one_hot(as.data.table(iris[, "Species"]))
head(binary_encoded)
#>    V1_setosa V1_versicolor V1_virginica
#> 1:         1             0            0
#> 2:         1             0            0
#> 3:         1             0            0
#> 4:         1             0            0
#> 5:         1             0            0
#> 6:         1             0            0
```

12.2.5 Base-N Encoding (Generalized Binary Encoding)

Expands Binary Encoding to base N instead of binary.

When to Use:

- Similar to Binary Encoding, but allows for greater flexibility.

12.2.6 Frequency Encoding

Replaces each category with its frequency (proportion) in the dataset.

Formula:

$$x_i' = \frac{\text{count}(x_i)}{\text{total count}}$$

When to Use:

- High-cardinality categorical variables.
- Feature engineering for boosting algorithms (e.g., LightGBM).

```
freq_encoding <- table(iris$Species) / length(iris$Species)
iris$Species_freq <-
    iris$Species %>% as.character() %>% map_dbl(~ freq_encoding[.])
head(iris$Species_freq)
#> [1] 0.3333333 0.3333333 0.3333333 0.3333333 0.3333333 0.3333333
```

12.2.7 Target Encoding (Mean Encoding)

Encodes categories using the mean of the target variable.

Formula:

$$x_i' = E[Y|X = x_i]$$

When to Use:

- Predictive models with categorical features strongly correlated with the target.
- High-cardinality categorical variables.
- Risk: Can lead to data leakage (use cross-validation).

```
library(data.table)
iris_dt <- as.data.table(iris)
iris_dt[, Species_mean := mean(Sepal.Length), by = Species]
head(iris_dt$Species_mean)
#> [1] 5.006 5.006 5.006 5.006 5.006 5.006
```

12.2.8 Ordinal Encoding

Maps categories to **ordered integer values** based on logical ranking.

Example:

Category	Ordinal Encoding
Low	1
Medium	2
High	3

When to Use:

- Ordinal variables with meaningful order (e.g., satisfaction ratings).

```
iris$Species_ordinal <-
    as.numeric(factor(iris$Species,
                      levels = c("setosa", "versicolor", "virginica")))
head(iris$Species_ordinal)
#> [1] 1 1 1 1 1 1
```

12.2.9 Weight of Evidence Encoding

- **Concept:**
  WoE is a method to convert categorical data into numerical values that capture the strength of the relationship between a feature (or category) and a binary outcome (like default vs. non-default).
- **The Formula:**
  $$\text{WoE} = \log\left(\frac{P(X_i|Y=1)}{P(X_i|Y=0)}\right)$$
  - $(X_i|Y=1)$: The probability (or proportion) of observing category X_i given the positive outcome (e.g., a "good" credit event).
  - $P(X_i|Y=0)$: The probability of observing category X_i given the negative outcome (e.g., a "bad" credit event).
  - **Logarithm:** Taking the log of the ratio gives us a symmetric scale where:
    * A positive WoE indicates the category is more associated with the positive outcome.
    * A negative WoE indicates the category is more associated with the negative outcome.

When and Why to Use WoE Encoding?

- **Logistic Regression in Credit Scoring:**
  Logistic regression models predict probabilities in terms of log-odds. WoE encoding aligns well with this because it essentially expresses how the odds of the positive outcome change with different categories. This is why it's popular in credit scoring models.
- **Interpretability:**
  The WoE transformation makes it easier to understand and interpret the relationship between each category of a variable and the outcome. Each category's WoE value tells you whether it increases or decreases the odds of a particular event (e.g., default).

Imagine you have a feature "Employment Status" with categories "Employed" and "Unemployed":

1. **Calculate Proportions:**
   - $P(\text{Employed}|Y = 1) = 0.8$ (80% of good credit cases are employed)
   - $P(\text{Employed}|Y = 0) = 0.4$ (40% of bad credit cases are employed)
2. **Compute WoE for "Employed":**

$$\text{WoE}_{\text{Employed}} = \log\left(\frac{0.8}{0.4}\right) = \log(2) \approx 0.693$$

   A positive value indicates that being employed increases the odds of a good credit outcome.
3. **Repeat for "Unemployed":**
   Suppose:
   - $P(\text{Unemployed}|Y = 1) = 0.2$
   - $P(\text{Unemployed}|Y = 0) = 0.6$

$$\text{WoE}_{\text{Unemployed}} = \log\left(\frac{0.2}{0.6}\right) = \log\left(\frac{1}{3}\right) \approx -1.099$$

   A negative value indicates that being unemployed is associated with a higher likelihood of a bad credit outcome.

Why is WoE Valuable?

- **Linear Relationship:**
  When you plug these WoE values into a logistic regression, the model essentially adds these values linearly, which fits nicely with how logistic regression models the log-odds.
- **Stability & Handling of Missing Values:**
  WoE can also help in smoothing out fluctuations in categorical data, especially when there are many levels or some levels with few observations.
- **Regulatory Acceptance:**
  In industries like banking, WoE is widely accepted because of its clear interpretability, which is crucial for compliance and transparency in credit risk modeling.

```r
# Load required packages
library(dplyr)
library(knitr)

# Create a sample dataset
# We assume 100 good credit cases and 100 bad credit cases
# Good credit: 80 "Employed" and 20 "Unemployed"
# Bad credit: 40 "Employed" and 60 "Unemployed"
data <- data.frame(
  employment_status = c(rep("Employed", 80), rep("Unemployed", 20),
```

```
                          rep("Employed", 40), rep("Unemployed", 60)),
  credit = c(rep(1, 100), rep(0, 100))
)

# Calculate counts for each category
woe_table <- data %>%
  group_by(employment_status) %>%
  summarise(
    good = sum(credit == 1),
    bad = sum(credit == 0)
  ) %>%
  # Calculate the distribution for good and bad credit cases
  mutate(
    dist_good = good / sum(good),
    dist_bad = bad / sum(bad),
    WoE = log(dist_good / dist_bad)
  )

# Print the WoE table
kable(woe_table)
```

employment_status	good	bad	dist_good	dist_bad	WoE
Employed	80	40	0.8	0.4	0.6931472
Unemployed	20	60	0.2	0.6	-1.0986123

```
# Merge the WoE values into the original data
data_woe <- data %>%
  left_join(woe_table %>% dplyr::select(employment_status, WoE), by = "employment_status")

head(data_woe)
#>   employment_status credit       WoE
#> 1          Employed      1 0.6931472
#> 2          Employed      1 0.6931472
#> 3          Employed      1 0.6931472
#> 4          Employed      1 0.6931472
#> 5          Employed      1 0.6931472
#> 6          Employed      1 0.6931472


# Fit a logistic regression model using WoE as predictor
model <- glm(credit ~ WoE, data = data_woe, family = binomial)

# Summary of the model
summary(model)
#>
```

```
#> Call:
#> glm(formula = credit ~ WoE, family = binomial, data = data_woe)
#>
#> Coefficients:
#>              Estimate Std. Error z value Pr(>|z|)
#> (Intercept) 1.023e-12  1.552e-01   0.000        1
#> WoE         1.000e+00  1.801e-01   5.552 2.83e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 277.26  on 199  degrees of freedom
#> Residual deviance: 242.74  on 198  degrees of freedom
#> AIC: 246.74
#>
#> Number of Fisher Scoring iterations: 4
```

When you fit a logistic regression using the WoE-encoded variable, the model is essentially:

$$\log\left(\frac{P(Y=1)}{P(Y=0)}\right) = \beta_0 + \beta_1 \cdot \text{WoE}$$

Here, **WoE** represents the Weight of Evidence value for a given category.

- **Log Odds Change:**
  β_1 indicates how much the log odds of a good credit outcome change for a one-unit increase in WoE. For example, if $\beta_1 = 0.5$, then a one-unit increase in WoE is associated with an increase of 0.5 in the log odds of having a good credit outcome.

- **Odds Ratio:**
  If you exponentiate β_1, you get the odds ratio. For instance, if $\beta_1 = 0.5$, then $\exp(0.5) \approx 1.65$. This means that for each one-unit increase in WoE, the odds of having a good credit outcome are multiplied by about 1.65.

Why is This Meaningful?

- **Direct Link to the Data:**
  The WoE value itself is a transformation of the original categorical variable that reflects the ratio of the proportions of good to bad outcomes for that category. By using WoE, you're directly incorporating this information into the model.

- **Interpretability:**
  The interpretation becomes intuitive:

  - A **positive** WoE indicates that the category is more associated with a good outcome.

- A **negative** WoE indicates that the category is more associated with a bad outcome.

Thus, if β_1 is positive, it suggests that as the category moves to one with a higher WoE (i.e., more favorable for a good outcome), the likelihood of a good outcome increases.

12.2.10 Helmert Encoding

Compares each category against the mean of previous categories.

When to Use:

- ANOVA models and categorical regression.

12.2.11 Probability Ratio Encoding

Encodes categories using the probability ratio of the target variable.

12.2.12 Backward Difference Encoding

Compares each category against the mean of all remaining categories.

12.2.13 Leave-One-Out Encoding

Similar to target encoding, but excludes the current observation to avoid bias.

12.2.14 James-Stein Encoding

A smoothed version of target encoding, reducing overfitting.

12.2.15 M-Estimator Encoding

Uses a Bayesian prior to smooth target encoding.

12.2.16 Thermometer Encoding

Similar to one-hot encoding, but retains ordinal structure.

12.2.17 Choosing the Right Encoding Method

Encoding Method	Best for Low Cardinality	Best for High Cardinality	Handles Ordinality	Suitable for Tree Models	Suitable for Linear Models
One-Hot Encoding	Yes	No	No	Yes	Yes

Encoding Method	Best for Low Cardinality	Best for High Cardinality	Handles Ordinality	Suitable for Tree Models	Suitable for Linear Models
Label Encoding	Yes	Yes	Yes	No	Yes
Target Encoding	Yes	Yes	No	Yes	Yes
Frequency Encoding	Yes	Yes	No	Yes	Yes
Binary Encoding	Yes	Yes	No	Yes	Yes

Chapter 13

Imputation (Missing Data)

13.1 Introduction to Missing Data

Missing data is a common problem in statistical analyses and data science, impacting the quality and reliability of insights derived from datasets. One widely used approach to address this issue is **imputation**, where missing data is replaced with *reasonable estimates*.

13.1.1 Types of Imputation

Imputation can be categorized into:

1. **Unit Imputation**: Replacing an entire missing observation (i.e., all features for a single data point are missing).
2. **Item Imputation**: Replacing missing values for specific variables (features) within a dataset.

While imputation offers a means to make use of incomplete datasets, it has historically been viewed skeptically. This skepticism arises from:

1. Frequent **misapplication** of imputation techniques, which can introduce significant **bias** to estimates.
2. Limited **applicability**, as imputation works well only under certain assumptions about the missing data mechanism and research objectives.

Biases in imputation can arise from various factors, including:

- **Imputation method**: The chosen method can influence the results and introduce biases.
- **Missing data mechanism**: The nature of the missing data—whether it is Missing Completely at Random (MCAR) or Missing at Random (MAR)—affects the accuracy of imputation.

- **Proportion of missing data**: The amount of missing data significantly impacts the reliability of the imputation.
- **Available information in the dataset**: Limited information reduces the robustness of the imputed values.

13.1.2 When and Why to Use Imputation

The appropriateness of imputation depends on the nature of the missing data and the research goal:

- **Missing Data in the Outcome Variable** (y): Imputation in such cases is generally problematic, as it can distort statistical models and lead to misleading conclusions. For example, imputing outcomes in regression or classification problems can alter the underlying relationship between the dependent and independent variables.
- **Missing Data in Predictive Variables** (x): Imputation is more commonly applied here, especially for **non-random missing data**. Properly handled, imputation can enable the use of incomplete datasets while minimizing bias.

13.1.2.1 Objectives of Imputation

The utility of imputation methods differs substantially depending on whether the goal of the analysis is *inference/explanation* or *prediction*. Each goal has distinct priorities and tolerances for bias, variance, and assumptions about the missing data mechanism:

13.1.2.1.1 Inference/Explanation In causal inference or explanatory analyses, the primary objective is to ensure valid statistical inference, emphasizing unbiased estimation of parameters and accurate representation of uncertainty. The treatment of missing data must align closely with the assumptions about the mechanism behind the missing data—whether it is Missing Completely at Random (MCAR), Missing at Random (MAR), or Missing Not at Random (MNAR):

- **Bias Sensitivity:** Inference analyses require that imputed data preserve the integrity of the relationships among variables. Poorly executed imputation can introduce bias, even when it addresses missingness superficially.
- **Variance and Confidence Intervals:** For inference, the quality of the standard errors, confidence intervals, and test statistics is critical. Naive imputation methods (e.g., mean imputation) often fail to appropriately reflect the uncertainty due to missingness, leading to overconfidence in parameter estimates.
- **Mechanism Considerations:** Imputation methods, such as multiple imputation (MI), attempt to generate values consistent with the observed

data distribution while accounting for missing data uncertainty. However, MI's performance depends heavily on the validity of the MAR assumption. If the missingness mechanism is MNAR and not addressed adequately, the imputed data could yield biased parameter estimates, undermining the purpose of inference.

13.1.2.1.2 Prediction In predictive modeling, the primary goal is to maximize model accuracy (e.g., minimizing mean squared error for continuous outcomes or maximizing classification accuracy). Here, the focus shifts to optimizing predictive performance rather than ensuring unbiased parameter estimates:

- **Loss of Information:** Missing data reduces the amount of usable information in a dataset. Imputation allows the model to leverage all available data, rather than excluding incomplete cases via listwise deletion, which can significantly reduce sample size and model performance.
- **Impact on Model Fit:** In predictive contexts, imputation can reduce standard errors of the predictions and stabilize model coefficients by incorporating plausible estimates for missing values.
- **Flexibility with Mechanism:** Predictive models are less sensitive to the missing data mechanism than inferential models, as long as the imputed values help reduce variability and align with patterns in the observed data. Methods like K-Nearest Neighbors (KNN), iterative imputation, or even machine learning models (e.g., random forests for imputation) can be valuable, regardless of strict adherence to MAR or MCAR assumptions.
- **Trade-offs:** Overimputation, where too much noise or complexity is introduced in the imputation process, can harm prediction by introducing artifacts that degrade model generalizability.

13.1.2.1.3 Key Takeaways The usefulness of imputation depends on whether the goal of the analysis is **inference** or **prediction**:

- **Inference/Explanation:** The primary concern is valid statistical inference, where biased estimates are unacceptable. Imputation is often of limited value for this purpose, as it may not address the underlying missing data mechanism appropriately (Rubin, 1996).
- **Prediction:** Imputation can be more useful in predictive modeling, as it reduces the loss of information from incomplete cases. By leveraging observed data, imputation can lower standard errors and improve model accuracy.

13.1.3 Importance of Missing Data Treatment in Statistical Modeling

Proper handling of missing data ensures:

- **Unbiased Estimates:** Avoiding distortions in parameter estimates.
- **Accurate Standard Errors:** Ensuring valid hypothesis testing and confidence intervals.
- **Adequate Statistical Power:** Maximizing the use of available data.

Ignoring or mishandling missing data can lead to:

1. **Bias:** Systematic errors in parameter estimates, especially under MAR or MNAR mechanisms.
2. **Loss of Power:** Reduced sample size leads to larger standard errors and weaker statistical significance.
3. **Misleading Conclusions:** Over-simplistic imputation methods (e.g., mean substitution) can distort relationships among variables.

13.1.4 Prevalence of Missing Data Across Domains

Missing data affects virtually all fields:

- **Business:** Non-responses in customer surveys, incomplete sales records, and transactional errors.
- **Healthcare:** Missing data in electronic health records (EHRs) due to incomplete patient histories or inconsistent data entry.
- **Social Sciences:** Non-responses or partial responses in large-scale surveys, leading to biased conclusions.

13.1.5 Practical Considerations for Imputation

- **Diagnostic Checks:** Always examine the patterns and mechanisms of missing data before applying imputation (Diagnosing the Missing Data Mechanism).
- **Model Selection:** Align the imputation method with the missing data mechanism and research goal.
- **Validation:** Assess the impact of imputation on results through sensitivity analyses or cross-validation.

13.2 Theoretical Foundations of Missing Data

13.2.1 Definition and Classification of Missing Data

Missing data refers to the absence of values for some variables in a dataset. The mechanisms underlying missingness significantly impact the validity of statistical analyses and the choice of handling methods. These mechanisms are classified into three categories:

- Missing Completely at Random (MCAR): The probability of missingness is independent of both observed and unobserved data. In other words, the missing data occur entirely at random and are unrelated to any values in the dataset.
- Missing at Random (MAR): The probability of missingness is related to the observed data but not to the missing data itself. This means that, after controlling for observed variables, the missingness is random.
- Missing Not at Random (MNAR): The probability of missingness depends on unobserved data or the missing values themselves. In this case, the missingness is related to the very information that is missing, making it the most challenging type to handle in analysis.

13.2.1.1 Missing Completely at Random (MCAR)

MCAR occurs when the probability of missingness is entirely random and unrelated to either observed or unobserved variables. Under this mechanism, missing data do not introduce bias in parameter estimates when ignored, although statistical efficiency is reduced due to the smaller sample size.

Mathematical Definition: The missingness is independent of all data, both observed and unobserved:

$$P(Y_{\text{missing}}|Y, X) = P(Y_{\text{missing}})$$

Characteristics of MCAR:

- Missingness is completely unrelated to both observed and unobserved data.
- Analyses remain unbiased even if missing data are ignored, though they may lack efficiency due to reduced sample size.
- The missing data points represent a random subset of the overall data.

Examples:

- A sensor randomly fails at specific time points, unrelated to environmental or operational conditions.
- Survey participants randomly omit responses to certain questions without any systematic pattern.

Methods for Testing MCAR:

1. **Little's MCAR Test:** A formal statistical test to assess whether data are MCAR. A significant result suggests deviation from MCAR.
2. **Mean Comparison Tests:**
   - T-tests or similar approaches compare observed and missing data groups on other variables. Significant differences indicate potential bias.
   - Failure to reject the null hypothesis of no difference does not confirm MCAR but suggests consistency with the MCAR assumption.

Handling MCAR:

Since MCAR data introduce no bias, they can be handled using the following techniques:

1. **Complete Case Analysis (Listwise Deletion):**
   - Analyses are performed only on cases with complete data. While unbiased under MCAR, this method reduces sample size and efficiency.
2. **Universal Singular Value Thresholding (USVT):**
   - This technique is effective for MCAR data recovery but can only recover the mean structure, not the entire true distribution (Chatterjee, 2015).
3. **SoftImpute:**
   - A matrix completion method useful for some missing data problems but less effective when missingness is not MCAR (Hastie et al., 2015).
4. **Synthetic Nearest Neighbor Imputation:**
   - A robust method for imputing missing data. While primarily designed for MCAR, it can also handle certain cases of missing not at random (MNAR) (Agarwal et al., 2023). Available on GitHub: syntheticNN.

Notes:

- The "missingness" on one variable can be correlated with the "missingness" on another variable without violating the MCAR assumption.
- Absence of evidence for bias (e.g., failing to reject a t-test) does not confirm that the data are MCAR.

13.2.1.2 Missing at Random (MAR)

Missing at Random (MAR) occurs when missingness depends on observed variables but not the missing values themselves. This mechanism assumes that observed data provide sufficient information to explain the missingness. In other words, there is a systematic relationship between the propensity of missing values and the observed data, but not the missing data.

Mathematical Definition:

The probability of missingness is conditional only on observed data:

$$P(Y_{\text{missing}}|Y, X) = P(Y_{\text{missing}}|X)$$

This implies that whether an observation is missing is unrelated to the missing values themselves but is related to the observed values of other variables.

Characteristics of MAR:

- Missingness is systematically related to observed variables.
- The propensity for a data point to be missing is not related to the missing data but is related to some of the observed data.
- Analyses must account for observed data to mitigate bias.

Examples:

- Women are less likely to disclose their weight, but their gender is recorded. In this case, weight is MAR.
- Missing income data is correlated with education, which is observed. For example, individuals with higher education levels might be less likely to reveal their income.

Challenges in MAR:

- MAR is weaker than Missing Completely at Random (MCAR).
- It is impossible to directly test for MAR. Evidence for MAR relies on domain expertise and indirect statistical checks rather than direct tests.

Handling MAR:

Common methods for handling MAR include:

- **Multiple Imputation by Chained Equations (MICE):** Iteratively imputes missing values based on observed data.
- **Maximum Likelihood Estimation:** Estimates model parameters directly while accounting for MAR assumptions.
- **Regression-Based Imputation:** Predicts missing values using observed covariates.

These methods assume that observed variables fully explain the missingness. Effective handling of MAR requires careful modeling and often domain-specific knowledge to validate the assumptions underlying the analysis.

13.2.1.3 Missing Not at Random (MNAR)

Missing Not at Random (MNAR) is the most complex missing data mechanism. Here, missingness depends on unobserved variables or the values of the missing data themselves. This makes MNAR particularly challenging, as ignoring this dependency introduces significant bias in analyses.

Mathematical Definition:

The probability of missingness depends on the missing values:

$$P(Y_{\text{missing}}|Y, X) \neq P(Y_{\text{missing}}|X)$$

Characteristics of MNAR:

- Missingness cannot be fully explained by observed data.
- The cause of missingness is directly related to the unobserved values.
- Ignoring MNAR introduces significant bias in parameter estimates, often leading to invalid conclusions.

Examples:

- High-income individuals are less likely to disclose their income, and income itself is unobserved.
- Patients with severe symptoms drop out of a clinical study, leaving their health outcomes unrecorded.

Challenges in MNAR:

- MNAR is the most difficult missingness mechanism to address because the missing data mechanism must be explicitly modeled.
- Identifying MNAR often requires domain knowledge and auxiliary information beyond the observed dataset.

Handling MNAR:

MNAR requires explicit modeling of the missingness mechanism. Common approaches include:

- **Heckman Selection Models:** These models explicitly account for the selection process leading to missing data, adjusting for potential bias (Heckman, 1976).

- **Instrumental Variables:** Variables predictive of missingness but unrelated to the outcome can be used to mitigate bias (Sun et al., 2018; Tchetgen Tchetgen and Wirth, 2017).

- **Pattern-Mixture Models**: These models separate the data into groups (patterns) based on missingness and model each group separately. They are particularly useful when the relationship between missingness and missing values is complex.

- **Sensitivity Analysis:** Examines how conclusions change under different assumptions about the missing data mechanism.

- **Use of Auxiliary Data**

  Auxiliary data refers to external data sources or variables that can help explain the missingness mechanism.

- **Surrogate Variables**: Adding variables that correlate with missing data can improve imputation accuracy and mitigate the MNAR challenge.
- **Linking External Datasets**: Merging datasets from different sources can provide additional context or predictors for missingness.
- **Applications in Business**: In marketing, customer demographics or transaction histories often serve as auxiliary data to predict missing responses in surveys.

Additionally, data collection strategies, such as follow-up surveys or targeted sampling, can help mitigate MNAR effects by collecting information that directly addresses the missingness mechanism. However, such approaches can be resource-intensive and require careful planning.

13.2.2 Missing Data Mechanisms

Mechanism	Missingness Depends On	Implications	Examples
MCAR	Neither observed nor missing data	No bias; simplest to handle; decreases efficiency due to data loss.	Random sensor failure.
MAR	Observed data only	Requires observed data to explain missingness; common assumption in imputation methods.	Gender-based missingness of weight.
MNAR	Missing data itself or unobserved variables	Requires explicit modeling of the missingness mechanism; significant bias if ignored.	High-income individuals not disclosing income.

13.2.3 Relationship Between Mechanisms and Ignorability

The concept of ignorability is central to determining whether the missingness process must be explicitly modeled. Ignorability impacts the choice of methods for handling missing data and whether the missing data mechanism can be safely disregarded or must be explicitly accounted for.

13.2.3.1 Ignorable Missing Data

Missing data is **ignorable** under the following conditions:

1. The missing data mechanism is MAR or MCAR.
2. The parameters governing the missing data process are unrelated to the parameters of interest in the analysis.

In cases of ignorable missing data, there is no need to model the missingness mechanism explicitly unless you aim to improve the efficiency or precision of parameter estimates. Common imputation techniques, such as multiple imputation or maximum likelihood estimation, rely on the assumption of ignorability to produce unbiased parameter estimates.

Practical Considerations for Ignorable Missingness

Even though ignorable mechanisms simplify analysis, researchers must rigorously assess whether the missingness mechanism meets the MAR or MCAR criteria. Violations can lead to biased results, even if unintentionally overlooked.

For example: A survey on income may assume MAR if missingness is associated with respondent age (observed variable) but not income itself (unobserved variable). However, if income directly influences nonresponse, the assumption of MAR is violated.

13.2.3.2 Non-Ignorable Missing Data

Missing data is **non-ignorable** when:

1. The missingness mechanism depends on the values of the missing data themselves or on unobserved variables.
2. The missing data mechanism is related to the parameters of interest, resulting in bias if the mechanism is not modeled explicitly.

This type of missingness (i.e., Missing Not at Random (MNAR) requires modeling the missing data mechanism directly to produce unbiased estimates.

Characteristics of Non-Ignorable Missingness

- **Dependence on Missing Values**: The likelihood of missingness is associated with the missing values themselves.
  - Example: In a study on health, individuals with more severe conditions are more likely to drop out, leading to an underrepresentation of the sickest individuals in the data.
- **Bias in Complete Case Analysis**: Analyses based solely on complete cases can lead to substantial bias.
  - Example: In income surveys, if wealthier individuals are less likely to report their income, the estimated mean income will be systematically lower than the true population mean.

- **Need for Explicit Modeling**: To address MNAR, the analyst must model the missing data mechanism. This often involves specifying relationships between observed data, missing data, and the missingness process itself.

13.2.3.3 Implications of Non-Ignorable Missingness

Non-ignorable mechanisms are often associated with sensitive or personal data:

- **Examples**:
  - Individuals with lower education levels may omit their education information.
  - Participants with controversial or stigmatized health conditions might opt out of surveys entirely.
- **Impact on Policy and Decision-Making**:
  - Biases introduced by MNAR can have serious consequences for policymaking, such as underestimating the prevalence of poverty or mischaracterizing population health needs.

By explicitly addressing non-ignorable missingness, researchers can mitigate biases and ensure that findings accurately reflect the underlying population.

13.3 Diagnosing the Missing Data Mechanism

Understanding the mechanism behind missing data is critical to choosing the appropriate methods for handling it. The three main mechanisms for missing data are **MCAR (Missing Completely at Random)**, **MAR (Missing at Random)**, and **MNAR (Missing Not at Random)**. This section discusses methods for diagnosing these mechanisms, including descriptive and inferential approaches.

13.3.1 Descriptive Methods

13.3.1.1 Visualizing Missing Data Patterns

Visualization tools are essential for detecting patterns in missing data. Heatmaps and correlation plots can help identify systematic missingness and provide insights into the underlying mechanism.

```
# Example: Visualizing missing data
library(Amelia)
missmap(
```

```
    airquality,
    main = "Missing Data Heatmap",
    col = c("yellow", "black"),
    legend = TRUE
)
```

- **Heatmaps**: Highlight where missingness occurs in a dataset.
- **Correlation Plots**: Show relationships between missingness indicators of different variables.

Exploring Univariate and Multivariate Missingness

- **Univariate Analysis**: Calculate the proportion of missing data for each variable.

```
# Example: Proportion of missing values
missing_proportions <- colSums(is.na(airquality)) / nrow(airquality)
print(missing_proportions)
#>      Ozone    Solar.R       Wind       Temp      Month        Day
#> 0.24183007 0.04575163 0.00000000 0.00000000 0.00000000 0.00000000
```

- **Multivariate Analysis**: Examine whether missingness in one variable is related to others. This can be visualized using scatterplots of observed vs. missing values.

```
# Example: Missingness correlation
library(naniar)
vis_miss(airquality)
```

```
gg_miss_upset(airquality) # Displays a missingness upset plot
```

13.3.2 Statistical Tests for Missing Data Mechanisms

13.3.2.1 Diagnosing MCAR: Little's Test

Little's test is a hypothesis test to determine if the missing data mechanism is **MCAR**. It tests whether the means of observed and missing data are significantly different. The null hypothesis is that the data are MCAR.

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

Where:

- O_i= Observed frequency
- E_i= Expected frequency under MCAR

```
# Example: Little's test
naniar::mcar_test(airquality)
#> # A tibble: 1 x 4
#>   statistic    df p.value missing.patterns
#>       <dbl> <dbl>   <dbl>            <int>
#> 1      35.1    14 0.00142                4
misty::na.test(airquality)
#>  Little's MCAR Test
#>
#>     n nIncomp nPattern  chi2 df  pval
#>   153      42        4 35.15 14 0.001
```

13.3.2.2 Diagnosing MCAR via Dummy Variables

Creating a binary indicator for missingness allows you to test whether the presence of missing data is related to observed data. For instance:

1. Create a dummy variable:
   - 1 = Missing
   - 0 = Observed
2. Conduct a chi-square test or t-test:
   - Chi-square: Compare proportions of missingness across groups.
   - T-test: Compare means of (other) observed variables with missingness indicators.

```
# Example: Chi-square test
airquality$missing_var <- as.factor(ifelse(is.na(airquality$Ozone), 1, 0))
# Across groups of months
table(airquality$missing_var, airquality$Month)
#>
#>      5  6  7  8  9
#>   0 26  9 26 26 29
#>   1  5 21  5  5  1
chisq.test(table(airquality$missing_var, airquality$Month))
#>
#>  Pearson's Chi-squared test
```

```
#>
#> data:  table(airquality$missing_var, airquality$Month)
#> X-squared = 44.751, df = 4, p-value = 4.48e-09

# Example: T-test (of other variable)
t.test(Wind ~ missing_var, data = airquality)
#>
#>  Welch Two Sample t-test
#>
#> data:  Wind by missing_var
#> t = -0.60911, df = 63.646, p-value = 0.5446
#> alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
#> 95 percent confidence interval:
#>  -1.6893132  0.8999377
#> sample estimates:
#> mean in group 0 mean in group 1
#>        9.862069       10.256757
```

13.3.3 Assessing MAR and MNAR

13.3.3.1 Sensitivity Analysis

Sensitivity analysis involves simulating different scenarios of missing data and assessing how the results change. For example, imputing missing values under different assumptions can provide insight into whether the data are MAR or MNAR.

13.3.3.2 Proxy Variables and External Data

Using proxy variables or external data sources can help assess whether missingness depends on unobserved variables (MNAR). For example, in surveys, follow-ups with non-respondents can reveal systematic differences.

13.3.3.3 Practical Challenges in Distinguishing MAR from MNAR

Distinguishing between Missing at Random (MAR) and Missing Not at Random (MNAR) is a critical and challenging task in data analysis. Properly identifying the nature of the missing data has significant implications for the choice of imputation strategies, model robustness, and the validity of conclusions. While statistical tests can sometimes aid in this determination, the process often relies heavily on domain knowledge, intuition, and exploratory analysis. Below, we discuss key considerations and examples that highlight these challenges:

- **Sensitive Topics**: Missing data related to sensitive or stigmatized topics, such as income, drug use, or health conditions, are often MNAR. For example, individuals with higher incomes might deliberately choose not to report their earnings due to privacy concerns. Similarly, participants

in a health survey may avoid answering questions about smoking if they perceive social disapproval. In such cases, the probability of missingness is directly related to the unobserved value itself, making MNAR likely.

- **Field-Specific Norms**: Understanding norms and typical data collection practices in a specific field can provide insights into missingness patterns. For instance, in marketing surveys, respondents may skip questions about spending habits if they consider the questions intrusive. Prior research or historical data from the same domain can help infer whether missingness is more likely MAR (e.g., random skipping due to survey fatigue) or MNAR (e.g., deliberate omission by higher spenders).

- **Analyzing Auxiliary Variables**: Leveraging auxiliary variables—those correlated with the missing variable—can help infer the missingness mechanism. For example, if missing income data strongly correlates with employment status, this suggests a MAR mechanism, as the missingness depends on observed variables. However, if missingness persists even after accounting for observable predictors, MNAR might be at play.

- **Experimental Design and Follow-Up**: In longitudinal studies, dropout rates can signal MAR or MNAR patterns. For example, if dropouts occur disproportionately among participants reporting lower satisfaction in early surveys, this indicates an MNAR mechanism. Designing follow-up surveys to specifically investigate dropout reasons can clarify missingness patterns.

- **Sensitivity Analysis**: To account for uncertainty in the missingness mechanism, researchers can conduct sensitivity analyses by comparing results under different assumptions (e.g., imputing data using both MAR and MNAR approaches). This process helps to quantify the potential impact of misclassifying the missingness mechanism on study conclusions.

- **Real-World Examples**:
  - In customer feedback surveys, higher ratings might be overrepresented due to non-response bias. Customers with negative experiences might be less likely to complete surveys, leading to an MNAR scenario.
  - In financial reporting, missing audit data might correlate with companies in financial distress, a classic MNAR case where the missingness depends on unobserved financial health metrics.

Summary

- **MCAR**: No pattern in missingness; use Little's test or dummy variable analysis.

- **MAR**: Missingness related to observed data; requires modeling assumptions or proxy analysis.

- **MNAR**: Missingness depends on unobserved data; requires external validation or sensitivity analysis.

13.4 Methods for Handling Missing Data

13.4.1 Basic Methods

13.4.1.1 Complete Case Analysis (Listwise Deletion)

Listwise deletion retains only cases with complete data for all features, discarding rows with any missing values.

Advantages:

- Universally applicable to various statistical tests (e.g., SEM, multilevel regression).
- When data are Missing Completely at Random (MCAR), parameter estimates and standard errors are unbiased.
- Under specific Missing at Random (MAR) conditions, such as when the probability of missing data depends only on independent variables, listwise deletion can still yield unbiased estimates. For instance, in the model $y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$, if missingness in X_1 is independent of y but depends on X_1 and X_2, the estimates remain unbiased (Little, 1992).
  - This aligns with principles of stratified sampling, which does not bias estimates.
  - In logistic regression, if missing data depend only on the dependent variable but not on independent variables, listwise deletion produces consistent slope estimates, though the intercept may be biased (Vach and Vach, 1994).
- For regression analysis, listwise deletion is more robust than Maximum Likelihood (ML) or Multiple Imputation (MI) when the MAR assumption is violated.

Disadvantages:

- Results in larger standard errors compared to advanced methods.
- If data are MAR but not MCAR, biased estimates can occur.
- In non-regression contexts, more sophisticated methods often outperform listwise deletion.

13.4.1.2 Available Case Analysis (Pairwise Deletion)

Pairwise deletion calculates estimates using all available data for each pair of variables, without requiring complete cases. It is particularly suitable for methods like linear regression, factor analysis, and SEM, which rely on correlation or covariance matrices.

Advantages:

- Under MCAR, pairwise deletion produces consistent and unbiased estimates in large samples.
- Compared to listwise deletion (Glasser, 1964):
  - When variable correlations are low, pairwise deletion provides more efficient estimates.
  - When correlations are high, listwise deletion becomes more efficient.

Disadvantages:

- Yields biased estimates under MAR conditions.
- In small samples, covariance matrices might not be positive definite, rendering coefficient estimation infeasible.
- Software implementation varies in how sample size is handled, potentially affecting standard errors.

Note: Carefully review software documentation to understand how sample size is treated, as this influences standard error calculations.

13.4.1.3 Indicator Method (Dummy Variable Adjustment)

Also known as the Missing Indicator Method, this approach introduces an additional variable to indicate missingness in the dataset.

Implementation:

1. Create an indicator variable:

$$D = \begin{cases} 1 & \text{if data on } X \text{ are missing} \\ 0 & \text{otherwise} \end{cases}$$

2. Modify the original variable to accommodate missingness:

$$X^* = \begin{cases} X & \text{if data are available} \\ c & \text{if data are missing} \end{cases}$$

Note: A common choice for c is the mean of X.

Interpretation:

- The coefficient of D represents the difference in the expected value of Y between cases with missing data and those without.
- The coefficient of X^* reflects the effect of X on Y for cases with observed data.

Disadvantages:

- Produces biased estimates of coefficients, even under MCAR conditions (Jones, 1996).
- May lead to overinterpretation of the "missingness effect," complicating model interpretation.

13.4.1.4 Advantages and Limitations of Basic Methods

Method	Advantages	Disadvantages
Listwise Deletion	Simple and universally applicable; unbiased under MCAR; robust in certain MAR scenarios.	Inefficient (larger standard errors); biased under MAR in many cases; discards potentially useful data.
Pairwise Deletion	Utilizes all available data; efficient under MCAR with low correlations; avoids discarding all cases.	Biased under MAR; prone to non-positive-definite covariance matrices in small samples.
Indicator Method	Simple implementation; explicitly models missingness effect.	Biased even under MCAR; complicates interpretation; may not reflect true underlying relationships.

13.4.2 Single Imputation Techniques

Single imputation methods replace missing data with a single value, generating a complete dataset that can be analyzed using standard techniques. While convenient, single imputation generally underestimates variability and risks biasing results.

13.4.2.1 Deterministic Methods

13.4.2.1.1 Mean, Median, Mode Imputation This method replaces missing values with the mean, median, or mode of the observed data.

Advantages:

- Simplicity and ease of implementation.
- Useful for quick exploratory data analysis.

Disadvantages:

- **Bias in Variances and Relationships**: Mean imputation reduces variance and disrupts relationships among variables, leading to biased estimates of variances and covariances (Haitovsky, 1968).

- **Underestimated Standard Errors**: Results in overly optimistic conclusions and increased risk of Type I errors.
- **Dependency Structure Ignored**: Particularly problematic in high-dimensional data, as it fails to capture dependencies among features.

13.4.2.1.2 Forward and Backward Filling (Time Series Contexts) Used in time series analysis, this method replaces missing values using the preceding (forward filling) or succeeding (backward filling) values.

Advantages:

- Simple and preserves temporal ordering.
- Suitable for datasets where adjacent values are strongly correlated.

Disadvantages:

- Biased if missingness spans long gaps or occurs systematically.
- Cannot capture trends or changes in the underlying process.

13.4.2.2 Statistical Prediction Models

13.4.2.2.1 Linear Regression Imputation Missing values in a variable are imputed based on a linear regression model using observed values of other variables.

Advantages:

- Preserves relationships between variables.
- More sophisticated than mean or median imputation.

Disadvantages:

- Assumes linear relationships, which may not hold in all datasets.
- Fails to capture variability, leading to downwardly biased standard errors.

13.4.2.2.2 Logistic Regression for Categorical Variables Similar to linear regression imputation but used for categorical variables. The missing category is predicted using a logistic regression model.

Advantages:

- Useful for binary or multinomial categorical data.
- Preserves relationships with other variables.

Disadvantages:

- Assumes the underlying logistic model is appropriate.
- Does not account for uncertainty in the imputed values.

13.4.2.3 Non-Parametric Methods

13.4.2.3.1 Hot Deck Imputation Hot Deck Imputation is a method of handling missing data where missing values are replaced with observed values from "donor" cases that are similar in other characteristics. This technique has been widely used in survey data, including by organizations like the U.S. Census Bureau, due to its flexibility and ability to maintain observed data distributions.

Advantages of Hot Deck Imputation

- **Retains observed data distributions**: Since missing values are imputed using actual observed data, the overall distribution remains realistic.
- **Flexible**: This method is applicable to both categorical and continuous variables.
- **Constrained imputations**: Imputed values are always feasible, as they come from observed cases.
- **Adds variability**: By randomly selecting donors, this method introduces variability, which can aid in robust standard error estimation.

Disadvantages of Hot Deck Imputation

- **Sensitivity to similarity definitions**: The quality of imputed values depends on the criteria used to define similarity between cases.
- **Computational intensity**: Identifying similar cases and randomly selecting donors can be computationally expensive, especially for large datasets.
- **Subjectivity**: Deciding how to define "similar" can introduce subjectivity or bias.

Algorithm for Hot Deck Imputation

Let n_1 represent the number of cases with complete data on the variable Y, and n_0 represent the number of cases with missing data on Y. The steps are as follows:

1. From the n_1 cases with complete data, take a random sample (with replacement) of n_1 cases.
2. From this sampled pool, take another random sample (with replacement) of size n_0.
3. Assign the values from the sampled n_0 cases to the cases with missing data in Y.
4. Repeat this process for every variable in the dataset.
5. For multiple imputation, repeat the above four steps multiple times to create multiple imputed datasets.

Variations and Considerations

- **Skipping Step 1**: If Step 1 is skipped, the variability of imputed values is reduced. This approach might not fully account for the uncertainty in

missing data, which can underestimate standard errors.

- **Defining similarity**: A major challenge in this method is deciding what constitutes "similarity" between cases. Common approaches include matching based on distance metrics (e.g., Euclidean distance) or grouping cases by strata or clusters.

Practical Example

The U.S. Census Bureau employs an approximate Bayesian bootstrap variation of Hot Deck Imputation. In this approach:

- Similar cases are identified based on shared characteristics or grouping variables.
- A randomly chosen value from a similar individual in the sample is used to replace the missing value.

This method ensures imputed values are plausible while incorporating variability.

Key Notes

- **Good aspects**:
  - Imputed values are constrained to observed possibilities.
  - Random selection introduces variability, helpful for multiple imputation scenarios.
- **Challenges**:
  - Defining and operationalizing "similarity" remains a critical step in applying this method effectively.

Below is an example code snippet illustrating Hot Deck Imputation in R:

```r
library(Hmisc)

# Example dataset with missing values
data <- data.frame(
  ID = 1:10,
  Age = c(25, 30, NA, 40, NA, 50, 60, NA, 70, 80),
  Gender = c("M", "F", "F", "M", "M", "F", "M", "F", "M", "F")
)

# Perform Hot Deck Imputation using Hmisc::impute
data$Age_imputed <- impute(data$Age, "random")

# Display the imputed dataset
print(data)
#>    ID Age Gender Age_imputed
#> 1   1  25      M          25
#> 2   2  30      F          30
#> 3   3  NA      F          25
```

```
#> 4   4  40      M          40
#> 5   5  NA      M          30
#> 6   6  50      F          50
#> 7   7  60      M          60
#> 8   8  NA      F          70
#> 9   9  70      M          70
#> 10 10  80      F          80
```

This code randomly imputes missing values in the `Age` column based on observed data using the `Hmisc` package's `impute` function.

13.4.2.3.2 Cold Deck Imputation Cold Deck Imputation is a systematic variant of Hot Deck Imputation where the donor pool is predefined. Instead of selecting donors dynamically from within the same dataset, Cold Deck Imputation relies on an external reference dataset, such as historical data or other high-quality external sources.

Advantages of Cold Deck Imputation

- **Utilizes high-quality external data**: This method is particularly useful when reliable external reference datasets are available, allowing for accurate and consistent imputations.
- **Consistency**: If the same donor pool is used across multiple datasets, imputations remain consistent, which can be advantageous in longitudinal studies or standardized processes.

Disadvantages of Cold Deck Imputation

- **Lack of adaptability**: External data may not adequately reflect the unique characteristics or variability of the current dataset.
- **Potential for systematic bias**: If the donor pool is significantly different from the target dataset, imputations may introduce bias.
- **Reduces variability**: Unlike Hot Deck Imputation, Cold Deck Imputation systematically selects values, which removes random variation. This can affect the estimation of standard errors and other inferential statistics.

Key Characteristics

- **Systematic Selection**: Cold Deck Imputation selects donor values systematically based on predefined rules or matching criteria, rather than using random sampling.
- **External Donor Pool**: Donors are typically drawn from a separate dataset or historical records.

Algorithm for Cold Deck Imputation

1. Identify an external reference dataset or predefined donor pool.
2. Define the matching criteria to find "similar" cases between the donor pool and the current dataset (e.g., based on covariates or stratification).

3. Systematically assign values from the donor pool to missing values in the current dataset based on the matching criteria.
4. Repeat the process for each variable with missing data.

Practical Considerations

- Cold Deck Imputation works well when external data closely resemble the target dataset. However, when there are significant differences in distributions or relationships between variables, imputations may be biased or unrealistic.

- This method is less useful for datasets without access to reliable external reference data.

Suppose we have a current dataset with missing values and a historical dataset with similar variables. The following example demonstrates how Cold Deck Imputation can be implemented:

```r
# Current dataset with missing values
current_data <- data.frame(
  ID = 1:5,
  Age = c(25, 30, NA, 45, NA),
  Gender = c("M", "F", "F", "M", "M")
)

# External reference dataset (donor pool)
reference_data <- data.frame(
  Age = c(28, 35, 42, 50),
  Gender = c("M", "F", "F", "M")
)

# Perform Cold Deck Imputation
library(dplyr)

# Define a matching function to find closest donor
impute_cold_deck <- function(missing_row, reference_data) {
  # Filter donors with the same gender
  possible_donors <- reference_data %>%
    filter(Gender == missing_row$Gender)

  # Return the mean age of matching donors as an example of systematic imputation
  return(mean(possible_donors$Age, na.rm = TRUE))
}

# Apply Cold Deck Imputation to the missing rows
current_data <- current_data %>%
  rowwise() %>%
```

```
  mutate(
    Age_imputed = ifelse(
      is.na(Age),
      impute_cold_deck(cur_data(), reference_data),
      Age
    )
  )

# Display the imputed dataset
print(current_data)
#> # A tibble: 5 x 4
#> # Rowwise:
#>      ID   Age Gender Age_imputed
#>   <int> <dbl> <chr>        <dbl>
#> 1     1    25 M             25
#> 2     2    30 F             30
#> 3     3    NA F             38.8
#> 4     4    45 M             45
#> 5     5    NA M             38.8
```

Comparison to Hot Deck Imputation

Feature	Hot Deck Imputation	Cold Deck Imputation
Donor Pool	Internal (within the dataset)	External (predefined dataset)
Selection	Random	Systematic
Variability	Retained	Reduced
Bias Potential	Lower	Higher (if donor pool differs)

This method suits situations where external reference datasets are trusted and representative. However, careful consideration is required to ensure alignment between the donor pool and the target dataset to avoid systematic biases.

13.4.2.3.3 Random Draw from Observed Distribution This imputation method replaces missing values by randomly sampling from the observed distribution of the variable with missing data. It is a simple, non-parametric approach that retains the variability of the original data.

Advantages

- **Preserves variability**:
  - By randomly drawing values from the observed data, this method ensures that the imputed values reflect the inherent variability of the variable.

- **Computational simplicity**:
  - The process is straightforward and does not require model fitting or complex calculations.

Disadvantages

- **Ignores relationships among variables**:
  - Since the imputation is based solely on the observed distribution of the variable, it does not consider relationships or dependencies with other variables.
- **May not align with trends**:
  - Imputed values are random and may fail to align with patterns or trends present in the data, such as time series structures or interactions.

Steps in Random Draw Imputation

1. Identify the observed (non-missing) values of the variable.
2. For each missing value, randomly sample one value from the observed distribution with or without replacement.
3. Replace the missing value with the randomly sampled value.

The following example demonstrates how to use random draw imputation to fill in missing values:

```
# Example dataset with missing values
set.seed(123)
data <- data.frame(
  ID = 1:10,
  Value = c(10, 20, NA, 30, 40, NA, 50, 60, NA, 70)
)

# Perform random draw imputation
random_draw_impute <- function(data, variable) {
  observed_values <- data[[variable]][!is.na(data[[variable]])] # Observed values
  data[[variable]][is.na(data[[variable]])] <- sample(observed_values,
                                                      sum(is.na(data[[variable]])),
                                                      replace = TRUE)
  return(data)
}

# Apply the imputation
imputed_data <- random_draw_impute(data, variable = "Value")

# Display the imputed dataset
print(imputed_data)
#>    ID Value
#> 1   1    10
```

```
#> 2    2     20
#> 3    3     70
#> 4    4     30
#> 5    5     40
#> 6    6     70
#> 7    7     50
#> 8    8     60
#> 9    9     30
#> 10  10     70
```

Considerations

- **When to Use**:
  - This method is suitable for exploratory analysis or as a quick way to handle missing data in univariate contexts.
- **Limitations**:
  - Random draws may result in values that do not fit well in the broader context of the dataset, especially in cases where the variable has strong relationships with others.

Feature	Random Draw from Observed Distribution	Regression-Based Imputation
Complexity	Simple	Moderate to High
Preserves Variability	Yes	Limited in deterministic forms
Considers Relationships	No	Yes
Risk of Implausible Values	Low (if observed values are plausible)	Moderate to High

This method is a quick and computationally efficient way to address missing data but is best complemented by more sophisticated methods when relationships between variables are important.

13.4.2.4 Semi-Parametric Methods

13.4.2.4.1 Predictive Mean Matching (PMM) Predictive Mean Matching (PMM) imputes missing values by finding observed values closest in predicted value (based on a regression model) to the missing data. The donor values are then used to fill in the gaps.

Advantages:

- Maintains observed variability in the data.

- Ensures imputed values are realistic since they are drawn from observed data.

Disadvantages:

- Requires a suitable predictive model.
- Computationally intensive for large datasets.

Steps for PMM:

1. Regress Y on X (matrix of covariates) for the n_1 (non-missing cases) to estimate coefficients $\hat{b}$ and residual variance s^2.
2. Draw from the posterior predictive distribution of residual variance:

$$s^2_{[1]} = \frac{(n_1 - k)s^2}{\chi^2},$$

   where χ^2 is a random draw from $\chi^2_{n_1-k}$.
3. Randomly sample from the posterior distribution of $\hat{b}$:

$$b_{[1]} \sim MVN(\hat{b}, s^2_{[1]}(X'X)^{-1}).$$

4. Standardize residuals for n_1 cases:

$$e_i = \frac{y_i - \hat{b}x_i}{\sqrt{s^2(1 - k/n_1)}}.$$

5. Randomly draw a sample (with replacement) of n_0 residuals from Step 4.
6. Calculate imputed values for n_0 missing cases:

$$y_i = b_{[1]}x_i + s_{[1]}e_i.$$

7. Repeat Steps 2–6 (except Step 4) to create multiple imputations.

Notes:

- PMM can handle heteroskedasticity
- works for multiple variables, imputing each using all others as predictors.

Example:

Example from Statistics Globe

```
set.seed(1) # Seed
N  <- 100                                       # Sample size
y  <- round(runif(N,-10, 10))                    # Target variable Y
x1 <- y + round(runif(N, 0, 50))                 # Auxiliary variable 1
x2 <- round(y + 0.25 * x1 + rnorm(N,-3, 15))     # Auxiliary variable 2
x3 <- round(0.1 * x1 + rpois(N, 2))              # Auxiliary variable 3
# (categorical variable)
```

```
x4 <- as.factor(round(0.02 * y + runif(N)))   # Auxiliary variable 4

# Insert 20% missing data in Y
y[rbinom(N, 1, 0.2) == 1] <- NA

data <- data.frame(y, x1, x2, x3, x4)         # Store data in dataset
head(data) # First 6 rows of our data
#>    y x1  x2 x3 x4
#> 1 NA 28 -10  5  0
#> 2 NA 15  -2  2  1
#> 3  1 15 -12  6  1
#> 4  8 58  22 10  1
#> 5 NA 26 -12  7  0
#> 6 NA 19  36  5  1

library("mice") # Load mice package

##### Impute data via predictive mean matching (single imputation)#####

imp_single <- mice(data, m = 1, method = "pmm") # Impute missing values
#>
#>  iter imp variable
#>   1   1  y
#>   2   1  y
#>   3   1  y
#>   4   1  y
#>   5   1  y
data_imp_single <- complete(imp_single)         # Store imputed data
# head(data_imp_single)

# Since single imputation underestiamtes stnadard errors,
# we use multiple imputaiton

##### Predictive mean matching (multiple imputation) #####

# Impute missing values multiple times
imp_multi <- mice(data, m = 5, method = "pmm")
#>
#>  iter imp variable
#>   1   1  y
#>   1   2  y
#>   1   3  y
#>   1   4  y
#>   1   5  y
#>   2   1  y
```

```
#>   2   2  y
#>   2   3  y
#>   2   4  y
#>   2   5  y
#>   3   1  y
#>   3   2  y
#>   3   3  y
#>   3   4  y
#>   3   5  y
#>   4   1  y
#>   4   2  y
#>   4   3  y
#>   4   4  y
#>   4   5  y
#>   5   1  y
#>   5   2  y
#>   5   3  y
#>   5   4  y
#>   5   5  y
data_imp_multi_all <-
    # Store multiply imputed data
    complete(imp_multi,
             "repeated",
             include = TRUE)

data_imp_multi <-
    # Combine imputed Y and X1-X4 (for convenience)
    data.frame(data_imp_multi_all[, 1:6], data[, 2:5])

head(data_imp_multi)
#>   y.0 y.1 y.2 y.3 y.4 y.5 x1  x2 x3 x4
#> 1  NA  -1   6  -1  -3   3 28 -10  5  0
#> 2  NA -10  10   4   0   2 15  -2  2  1
#> 3   1   1   1   1   1   1 15 -12  6  1
#> 4   8   8   8   8   8   8 58  22 10  1
#> 5  NA   0  -1  -6   2   0 26 -12  7  0
#> 6  NA   4   0   3   3   3 19  36  5  1
```

Example from UCLA Statistical Consulting

```
library(mice)
library(VIM)
library(lattice)
library(ggplot2)
## set observations to NA
anscombe <- within(anscombe, {
```

```
    y1[1:3] <- NA
    y4[3:5] <- NA
})
## view
head(anscombe)
#>   x1 x2 x3 x4   y1   y2    y3   y4
#> 1 10 10 10  8   NA 9.14  7.46 6.58
#> 2  8  8  8  8   NA 8.14  6.77 5.76
#> 3 13 13 13  8   NA 8.74 12.74   NA
#> 4  9  9  9  8 8.81 8.77  7.11   NA
#> 5 11 11 11  8 8.33 9.26  7.81   NA
#> 6 14 14 14  8 9.96 8.10  8.84 7.04

## check missing data patterns
md.pattern(anscombe)
```

```
#>   x1 x2 x3 x4 y2 y3 y1 y4
#> 6  1  1  1  1  1  1  1  1 0
#> 2  1  1  1  1  1  1  1  0 1
#> 2  1  1  1  1  1  1  0  1 1
#> 1  1  1  1  1  1  1  0  0 2
#>    0  0  0  0  0  0  3  3 6

## Number of observations per patterns for all pairs of variables
p <- md.pairs(anscombe)
p
#> $rr
```

```
#>    x1 x2 x3 x4 y1 y2 y3 y4
#> x1 11 11 11 11  8 11 11  8
#> x2 11 11 11 11  8 11 11  8
#> x3 11 11 11 11  8 11 11  8
#> x4 11 11 11 11  8 11 11  8
#> y1  8  8  8  8  8  8  8  6
#> y2 11 11 11 11  8 11 11  8
#> y3 11 11 11 11  8 11 11  8
#> y4  8  8  8  8  6  8  8  8
#>
#> $rm
#>    x1 x2 x3 x4 y1 y2 y3 y4
#> x1  0  0  0  0  3  0  0  3
#> x2  0  0  0  0  3  0  0  3
#> x3  0  0  0  0  3  0  0  3
#> x4  0  0  0  0  3  0  0  3
#> y1  0  0  0  0  0  0  0  2
#> y2  0  0  0  0  3  0  0  3
#> y3  0  0  0  0  3  0  0  3
#> y4  0  0  0  0  2  0  0  0
#>
#> $mr
#>    x1 x2 x3 x4 y1 y2 y3 y4
#> x1  0  0  0  0  0  0  0  0
#> x2  0  0  0  0  0  0  0  0
#> x3  0  0  0  0  0  0  0  0
#> x4  0  0  0  0  0  0  0  0
#> y1  3  3  3  3  0  3  3  2
#> y2  0  0  0  0  0  0  0  0
#> y3  0  0  0  0  0  0  0  0
#> y4  3  3  3  3  2  3  3  0
#>
#> $mm
#>    x1 x2 x3 x4 y1 y2 y3 y4
#> x1  0  0  0  0  0  0  0  0
#> x2  0  0  0  0  0  0  0  0
#> x3  0  0  0  0  0  0  0  0
#> x4  0  0  0  0  0  0  0  0
#> y1  0  0  0  0  3  0  0  1
#> y2  0  0  0  0  0  0  0  0
#> y3  0  0  0  0  0  0  0  0
#> y4  0  0  0  0  1  0  0  3
```

- `rr` = number of observations where both pairs of values are observed
- `rm` = the number of observations where both variables are missing values
- `mr` = the number of observations where the first variable's value (e.g. the

row variable) is observed and second (or column) variable is missing
- `mm` = the number of observations where the second variable's value (e.g. the col variable) is observed and first (or row) variable is missing

```
## Margin plot of y1 and y4
marginplot(anscombe[c(5, 8)], col = c("blue", "red", "orange"))
```

```
## 5 imputations for all missing values
imp1 <- mice(anscombe, m = 5)
#>
#>  iter imp variable
#>   1   1  y1  y4
#>   1   2  y1  y4
#>   1   3  y1  y4
#>   1   4  y1  y4
#>   1   5  y1  y4
#>   2   1  y1  y4
#>   2   2  y1  y4
#>   2   3  y1  y4
#>   2   4  y1  y4
#>   2   5  y1  y4
#>   3   1  y1  y4
#>   3   2  y1  y4
#>   3   3  y1  y4
#>   3   4  y1  y4
#>   3   5  y1  y4
#>   4   1  y1  y4
```

```
#>   4   2  y1  y4
#>   4   3  y1  y4
#>   4   4  y1  y4
#>   4   5  y1  y4
#>   5   1  y1  y4
#>   5   2  y1  y4
#>   5   3  y1  y4
#>   5   4  y1  y4
#>   5   5  y1  y4

## linear regression for each imputed data set - 5 regression are run
fitm <- with(imp1, lm(y1 ~ y4 + x1))
summary(fitm)
#> # A tibble: 15 x 6
#>    term        estimate std.error statistic p.value  nobs
#>    <chr>          <dbl>     <dbl>     <dbl>   <dbl> <int>
#>  1 (Intercept)    7.33      2.44       3.01  0.0169    11
#>  2 y4            -0.416     0.223     -1.86  0.0996    11
#>  3 x1             0.371     0.141      2.63  0.0302    11
#>  4 (Intercept)    7.27      2.90       2.51  0.0365    11
#>  5 y4            -0.435     0.273     -1.59  0.150     11
#>  6 x1             0.387     0.160      2.41  0.0422    11
#>  7 (Intercept)    6.54      2.80       2.33  0.0479    11
#>  8 y4            -0.322     0.255     -1.26  0.243     11
#>  9 x1             0.362     0.156      2.32  0.0491    11
#> 10 (Intercept)    5.93      3.08       1.92  0.0907    11
#> 11 y4            -0.286     0.282     -1.02  0.339     11
#> 12 x1             0.418     0.176      2.37  0.0451    11
#> 13 (Intercept)    8.16      2.67       3.05  0.0158    11
#> 14 y4            -0.489     0.251     -1.95  0.0867    11
#> 15 x1             0.326     0.151      2.17  0.0622    11

## pool coefficients and standard errors across all 5 regression models
pool(fitm)
#> Class: mipo    m = 5
#>          term m   estimate       ubar           b          t dfcom       df
#> 1 (Intercept) 5  7.0445966 7.76794670 0.719350800 8.63116766     8 5.805314
#> 2          y4 5 -0.3896685 0.06634920 0.006991497 0.07473900     8 5.706243
#> 3          x1 5  0.3727865 0.02473847 0.001134293 0.02609962     8 6.178032
#>          riv     lambda       fmi
#> 1 0.11112601 0.10001207 0.3044313
#> 2 0.12644909 0.11225460 0.3161877
#> 3 0.05502168 0.05215218 0.2586992

## output parameter estimates
```

```r
summary(pool(fitm))
#>          term   estimate std.error statistic       df    p.value
#> 1 (Intercept)  7.0445966 2.9378849  2.397846 5.805314 0.05483678
#> 2          y4 -0.3896685 0.2733843 -1.425350 5.706243 0.20638512
#> 3          x1  0.3727865 0.1615538  2.307508 6.178032 0.05923999
```

13.4.2.4.2 Stochastic Imputation Stochastic Imputation is an enhancement of regression imputation that introduces randomness into the imputation process by adding a random residual to the predicted values from a regression model. This approach aims to retain the variability of the original data while reducing the bias introduced by deterministic regression imputation.

Stochastic Imputation can be described as:

$$\text{Imputed Value} = \text{Predicted Value (from regression)} + \text{Random Residual}$$

This method is commonly used as a foundation for multiple imputation techniques.

Advantages of Stochastic Imputation

- **Retains all the benefits of regression imputation**:
  - Preserves relationships between variables in the dataset.
  - Utilizes information from observed data to inform imputations.
- **Introduces randomness**:
  - Adds variability by including a random residual term, making imputed values more realistic and better representing the uncertainty of missing data.
- **Supports multiple imputation**:
  - By generating different random residuals for each iteration, it facilitates the creation of multiple plausible datasets for robust statistical analysis.

Disadvantages of Stochastic Imputation

- **Implausible values**:
  - Depending on the random residuals, imputed values may fall outside the plausible range (e.g., negative values for variables like age or income).
- **Cannot handle heteroskedasticity**:
  - If the data exhibit heteroskedasticity (i.e., non-constant variance of residuals), the randomness added by stochastic imputation may not accurately reflect the underlying variability.

Steps in Stochastic Imputation

1. Fit a regression model using cases with complete data for the variable with missing values.
2. Predict missing values using the fitted model.
3. Generate random residuals based on the distribution of residuals from the regression model.
4. Add the random residuals to the predicted values to impute missing values.

```
# Example dataset with missing values
set.seed(123)
data <- data.frame(
  X = rnorm(10, mean = 50, sd = 10),
  Y = c(100, 105, 110, NA, 120, NA, 130, 135, 140, NA)
)

# Perform stochastic imputation
stochastic_impute <- function(data, predictor, target) {
  # Subset data with complete cases
  complete_data <- data[!is.na(data[[target]]), ]

  # Fit a regression model
  model <- lm(as.formula(paste(target, "~", predictor)), data = complete_data)

  # Predict missing values
  missing_data <- data[is.na(data[[target]]), ]
  predictions <- predict(model, newdata = missing_data)

  # Add random residuals
  residual_sd <- sd(model$residuals, na.rm = TRUE)
  stochastic_values <- predictions + rnorm(length(predictions), mean = 0, sd = residual

  # Impute missing values
  data[is.na(data[[target]]), target] <- stochastic_values
  return(data)
}

# Apply stochastic imputation
imputed_data <- stochastic_impute(data, predictor = "X", target = "Y")

# Display the imputed dataset
print(imputed_data)
```

Notes

- **Multiple Imputation**: Most multiple imputation methods are extensions of stochastic regression imputation. By repeating the imputation process with different random seeds, multiple datasets can be generated to account for uncertainty in the imputed values.

- **Dealing with Implausible Values**: Additional constraints or transformations (e.g., truncating imputed values to a plausible range) may be necessary to address the issue of implausible values.

Comparison to Deterministic Regression Imputation

Feature	Deterministic Regression Imputation	Stochastic Imputation
Randomness	None	Adds random residuals
Preserves Variability	No	Yes
Use in Multiple Imputation	Limited	Well-suited
Bias Potential	Higher	Lower

```
# Income data
set.seed(1)                                # Set seed
N <- 1000                                      # Sample size

income <-
  round(rnorm(N, 0, 500))             # Create some synthetic income data
income[income < 0] <- income[income < 0] * (-1)

x1 <- income + rnorm(N, 1000, 1500)            # Auxiliary variables
x2 <- income + rnorm(N,-5000, 2000)


# Create 10% missingness in income
income[rbinom(N, 1, 0.1) == 1] <- NA

data_inc_miss <- data.frame(income, x1, x2)
```

Single stochastic regression imputation

```
imp_inc_sri  <- mice(data_inc_miss, method = "norm.nob", m = 1)
#>
#>  iter imp variable
#>   1   1  income
#>   2   1  income
#>   3   1  income
#>   4   1  income
#>   5   1  income
data_inc_sri <- complete(imp_inc_sri)
```

Single predictive mean matching

```
imp_inc_pmm  <- mice(data_inc_miss, method = "pmm", m = 1)
#>
#>  iter imp variable
#>   1   1  income
#>   2   1  income
#>   3   1  income
#>   4   1  income
#>   5   1  income
data_inc_pmm <- complete(imp_inc_pmm)
```

Stochastic regression imputation contains negative values

```
data_inc_sri$income[data_inc_sri$income < 0]
#>  [1]  -23.85404  -58.37790  -61.86396  -57.47909  -21.29221  -73.26549
#>  [7]  -61.76194  -42.45942 -351.02991 -317.69090
# No values below 0
data_inc_pmm$income[data_inc_pmm$income < 0]
#> numeric(0)
```

Evidence for heteroskadastic data

```
# Heteroscedastic data

set.seed(1)                              # Set seed
N <- 1:1000                                   # Sample size

a <- 0
b <- 1
sigma2 <- N^2
eps <- rnorm(N, mean = 0, sd = sqrt(sigma2))

y <- a + b * N + eps                          # Heteroscedastic variable
x <- 30 * N + rnorm(N[length(N)], 1000, 200) # Correlated variable

y[rbinom(N[length(N)], 1, 0.3) == 1] <- NA   # 30% missing

data_het_miss <- data.frame(y, x)
```

Single stochastic regression imputation

```
imp_het_sri  <- mice(data_het_miss, method = "norm.nob", m = 1)
#>
#>  iter imp variable
#>   1   1  y
#>   2   1  y
#>   3   1  y
#>   4   1  y
```

```
#>   5   1  y
data_het_sri <- complete(imp_het_sri)
```

Single predictive mean matching

```
imp_het_pmm  <- mice(data_het_miss, method = "pmm", m = 1)
#>
#>  iter imp variable
#>   1   1  y
#>   2   1  y
#>   3   1  y
#>   4   1  y
#>   5   1  y
data_het_pmm <- complete(imp_het_pmm)
```

Comparison between predictive mean matching and stochastic regression imputation

```
par(mfrow = c(1, 2))                                # Both plots in one graphic

# Plot of observed values
plot(x[!is.na(data_het_sri$y)],
     data_het_sri$y[!is.na(data_het_sri$y)],
     main = "",
     xlab = "X",
     ylab = "Y")
# Plot of missing values
points(x[is.na(y)], data_het_sri$y[is.na(y)],
       col = "red")

# Title of plot
title("Stochastic Regression Imputation",
      line = 0.5)

# Regression line
abline(lm(y ~ x, data_het_sri),
       col = "#1b98e0", lwd = 2.5)

# Legend
legend(
  "topleft",
  c("Observed Values", "Imputed Values", "Regression Y ~ X"),
  pch = c(1, 1, NA),
  lty = c(NA, NA, 1),
  col = c("black", "red", "#1b98e0")
)
```

```r
# Plot of observed values
plot(x[!is.na(data_het_pmm$y)],
     data_het_pmm$y[!is.na(data_het_pmm$y)],
     main = "",
     xlab = "X",
     ylab = "Y")


# Plot of missing values
points(x[is.na(y)], data_het_pmm$y[is.na(y)],
       col = "red")

# Title of plot
title("Predictive Mean Matching",
      line = 0.5)
abline(lm(y ~ x, data_het_pmm),
       col = "#1b98e0", lwd = 2.5)

# Legend
legend(
  "topleft",
  c("Observed Values", "Imputed Values", "Regression Y ~ X"),
  pch = c(1, 1, NA),
  lty = c(NA, NA, 1),
  col = c("black", "red", "#1b98e0")
)

mtext(
  "Imputation of Heteroscedastic Data",
  # Main title of plot
  side = 3,
  line = -1.5,
  outer = TRUE,
  cex = 2
)
```

13.4.2.5 Matrix Completion

Matrix completion is a method used to impute missing data in a feature matrix while accounting for dependence between features. This approach leverages principal components to approximate the data matrix, a process referred to as **matrix completion** (James et al., 2013, Sec 12.3).

Problem Setup

Consider an $n \times p$ feature matrix $\mathbf{X}$, where the element x_{ij} represents the value for the ith observation and jth feature. Some elements of $\mathbf{X}$ are missing, and we aim to impute these missing values.

Similar to the process described in 25.2, the matrix $\mathbf{X}$ can be approximated using its leading principal components. Specifically, we consider M principal components that minimize the following objective:

$$\min_{\mathbf{A}\in\mathbb{R}^{n\times M},\mathbf{B}\in\mathbb{R}^{p\times M}} \left\{ \sum_{(i,j)\in\mathcal{O}} (x_{ij} - \sum_{m=1}^{M} a_{im} b_{jm})^2 \right\}$$

where $\mathcal{O}$ is the set of observed indices (i, j), which is a subset of the total $n \times p$ pairs. Here: - $\mathbf{A}$ is an $n \times M$ matrix of principal component scores. - $\mathbf{B}$ is a $p \times M$ matrix of principal component loadings.

Imputation of Missing Values

After solving the minimization problem:

1. Missing observations x_{ij} can be imputed using the formula:

$$\hat{x}_{ij} = \sum_{m=1}^{M} \hat{a}_{im} \hat{b}_{jm}$$

   where $\hat{a}_{im}$ and $\hat{b}_{jm}$ are the estimated elements of $\mathbf{A}$ and $\mathbf{B}$, respectively.
2. The leading M principal component scores and loadings can be approximately recovered, as is done in complete data scenarios.

Iterative Algorithm

The eigen-decomposition used in standard principal component analysis is not applicable here because of missing values. Instead, an iterative algorithm, as described in (James et al., 2013, Alg 12.1), is employed:

1. **Initialize the Complete Matrix**: Construct an initial complete matrix $\tilde{\mathbf{X}}$ of dimension $n \times p$ where:

$$\tilde{x}_{ij} = \begin{cases} x_{ij} & \text{if } (i,j) \in \mathcal{O} \\ \bar{x}_j & \text{if } (i,j) \notin \mathcal{O} \end{cases}$$

   Here, $\bar{x}_j$ is the mean of the observed values for the jth variable in the incomplete data matrix $\mathbf{X}$. $\mathcal{O}$ indexes the observed elements of $\mathbf{X}$.

2. **Iterative Steps**: Repeat the following steps until convergence:

   - **Minimize the Objective**: Solve the problem:

$$\min_{\mathbf{A} \in R^{n \times M}, \mathbf{B} \in R^{p \times M}} \left\{ \sum_{(i,j) \in \mathcal{O}} (x_{ij} - \sum_{m=1}^{M} a_{im} b_{jm})^2 \right\}$$

     by computing the principal components of the current $\tilde{\mathbf{X}}$.

   - **Update Missing Values**: For each missing element $(i,j) \notin \mathcal{O}$, set:

$$\tilde{x}_{ij} \leftarrow \sum_{m=1}^{M} \hat{a}_{im} \hat{b}_{jm}$$

   - **Recalculate the Objective**: Compute the objective:

$$\sum_{(i,j) \in \mathcal{O}} (x_{ij} - \sum_{m=1}^{M} \hat{a}_{im} \hat{b}_{jm})^2$$

3. **Return Imputed Values**: Once the algorithm converges, return the estimated missing entries $\tilde{x}_{ij}$ for $(i,j) \notin \mathcal{O}$.

Key Considerations

- This approach assumes that the missing data are missing at random (MAR).
- Convergence criteria for the iterative algorithm often involve achieving a threshold for the change in the objective function or limiting the number of iterations.
- The choice of M, the number of principal components, can be guided by cross-validation or other model selection techniques.

13.4.2.6 Comparison of Single Imputation Techniques

Method	Advantages	Disadvantages
Mean, Median, Mode Imputation	Simple, quick implementation.	Biased variances and covariances; ignores relationships among variables.
Forward/Backward Filling	Preserves temporal ordering.	Biased for systematic gaps or long missing sequences.
Linear Regression Imputation	Preserves relationships among variables.	Fails to capture variability; assumes linearity.
Logistic Regression Imputation	Handles categorical variables well.	Requires appropriate model assumptions; ignores variability.
PMM	Maintains variability; imputes realistic values.	Computationally intensive; requires a good predictive model.
Hot Deck Imputation	Flexible; maintains data distribution.	Sensitive to donor selection; computationally demanding.
Cold Deck Imputation	Consistent across datasets with predefined donor pools.	Risk of bias if donor data are not representative.
Random Draw from Observed	Simple; retains variability in data.	Does not preserve relationships among variables; random imputation may distort trends.
Matrix Completion	Captures dependencies; imputes structurally consistent values.	Computationally intensive; assumes principal components capture data relationships.

Single imputation techniques are straightforward and accessible, but they often underestimate uncertainty and fail to fully leverage relationships among variables. These limitations make them less ideal for rigorous analyses compared to multiple imputation or model-based approaches.

13.4.3 Machine Learning and Modern Approaches

13.4.3.1 Tree-Based Methods

13.4.3.1.1 Random Forest Imputation (missForest) Random Forest Imputation uses an iterative process where a random forest model predicts missing values for one variable at a time, treating other variables as predictors. This process continues until convergence.

- **Mathematical Framework**:
  1. For a variable X_j with missing values, treat X_j as the response variable.
  2. Fit a random forest model $f(X_{-j})$ using the other variables X_{-j} as predictors.
  3. Predict missing values $\hat{X}_j = f(X_{-j})$.
  4. Repeat for all variables with missing data until imputed values stabilize.
- **Advantages**:
  - Captures complex interactions and non-linearities.
  - Handles mixed data types seamlessly.
- **Limitations**:
  - Computationally intensive for large datasets.
  - Sensitive to the quality of data relationships.

13.4.3.1.2 Gradient Boosting Machines (GBM) Gradient Boosting Machines iteratively build models to minimize loss functions. For imputation, missing values are treated as a target variable to be predicted.

- **Mathematical Framework**: The GBM algorithm minimizes the loss function:

$$L = \sum_{i=1}^{n} \ell(y_i, f(x_i)),$$

  where ℓ is the loss function (e.g., mean squared error), y_i are observed values, and $f(x_i)$ are predictions.

- Missing values are treated as the y_i and predicted iteratively.

- **Advantages**:

  - Highly accurate predictions.
  - Captures variable importance.

- **Limitations**:

  - Overfitting risks.
  - Requires careful parameter tuning.

13.4.3.2 Neural Network-Based Imputation

13.4.3.2.1 Autoencoders Autoencoders are unsupervised neural networks that compress and reconstruct data. Missing values are estimated during reconstruction.

- **Mathematical Framework**: An autoencoder consists of:
  1. An encoder function: $h = g(Wx + b)$, which compresses the input x.
  2. A decoder function: $\hat{x} = g'(W'h + b')$, which reconstructs the data.
- The network minimizes the reconstruction loss:

$$L = \sum_{i=1}^{n} (x_i - \hat{x}_i)^2.$$

- **Advantages**:
  - Handles high-dimensional and non-linear data.
  - Unsupervised learning.
- **Limitations**:
  - Computationally demanding.
  - Requires large datasets for effective training.

13.4.3.2.2 Generative Adversarial Networks (GANs) for Data Imputation GANs consist of a generator and a discriminator. For imputation, the generator fills in missing values, and the discriminator evaluates the quality of the imputations.

- **Mathematical Framework**: GAN training involves optimizing:

$$\min_G \max_D \mathbb{E}[\log D(x)] + \mathbb{E}[\log(1 - D(G(z)))].$$

  - $D(x)$: Discriminator's probability that x is real.
  - $G(z)$: Generator's output for latent input z.
- **Advantages**:
  - Realistic imputations that reflect underlying distributions.
  - Handles complex data types.
- **Limitations**:
  - Difficult to train and tune.
  - Computationally intensive.

13.4.3.3 Matrix Factorization and Matrix Completion

13.4.3.3.1 Singular Value Decomposition (SVD) SVD decomposes a matrix A into three matrices:

$$A = U\Sigma V^T,$$

where U and V are orthogonal matrices, and Σ contains singular values. Missing values are estimated by reconstructing A using a low-rank approximation:

$$\hat{A} = U_k \Sigma_k V_k^T.$$

- **Advantages**:
  - Captures global patterns.
  - Efficient for structured data.
- **Limitations**:
  - Assumes linear relationships.
  - Sensitive to sparsity.

13.4.3.3.2 Collaborative Filtering Approaches Collaborative filtering uses similarities between rows (users) or columns (items) to impute missing data. For instance, the value of X_{ij} is predicted as:

$$\hat{X}_{ij} = \frac{\sum_{k \in N(i)} w_{ik} X_{kj}}{\sum_{k \in N(i)} w_{ik}},$$

where w_{ik} represents similarity weights and $N(i)$ is the set of neighbors.

13.4.3.4 K-Nearest Neighbor (KNN) Imputation

KNN identifies the k nearest observations based on a distance metric and imputes missing values using a weighted average (continuous variables) or mode (categorical variables).

- **Mathematical Framework**: For a missing value x, its imputed value is:

$$\hat{x} = \frac{\sum_{i=1}^{k} w_i x_i}{\sum_{i=1}^{k} w_i},$$

  where $w_i = \frac{1}{d(x, x_i)}$ and $d(x, x_i)$ is a distance metric (e.g., Euclidean or Manhattan).

- **Advantages**:
  - Simple and interpretable.

  - Non-parametric.

- **Limitations**:
  - Computationally expensive for large datasets.

13.4.3.5 Hybrid Methods

Hybrid methods combine statistical and machine learning approaches. For example, mean imputation followed by fine-tuning with machine learning models. These methods aim to leverage the strengths of multiple techniques.

13.4.3.6 Summary Table

Method	Advantages	Limitations	Applications
Random Forest (missForest)	Handles mixed data types, captures interactions	Computationally intensive	Mixed data types
Gradient Boosting Machines	High accuracy, feature importance	Sensitive to parameters	Predictive tasks
Autoencoders	Handles high-dimensional, non-linear data	Computationally expensive	Complex datasets
GANs	Realistic imputations, complex distributions	Difficult to train, resource-intensive	Healthcare, finance
SVD	Captures global patterns, efficient	Assumes linear relationships	Recommendation systems
Collaborative Filtering	Intuitive for user-item data	Struggles with sparse or new data	Recommender systems
KNN Imputation	Simple, interpretable	Computationally intensive, sensitive to k	General-purpose
Hybrid Methods	Combines multiple strengths	Complexity in design	Flexible

13.4.4 Multiple Imputation

Multiple Imputation (MI) is a statistical technique for handling missing data by creating several plausible datasets through imputation, analyzing each dataset separately, and then combining the results to account for uncertainty in the imputations. MI operates under the assumption that missing data is either Missing Completely at Random (MCAR) or Missing at Random (MAR).

Unlike Single Imputation Techniques, MI reflects the uncertainty inherent in the missing data by introducing variability in the imputed values. It avoids biases introduced by ad hoc methods and produces more reliable statistical inferences.

The three fundamental steps in MI are:

1. **Imputation**: Replace missing values with a set of plausible values to create multiple "completed" datasets.
2. **Analysis**: Perform the desired statistical analysis on each imputed dataset.
3. **Combination**: Combine the results using rules to account for within- and between-imputation variability.

13.4.4.1 Why Multiple Imputation is Important

Imputed values are estimates and inherently include random error. However, when these estimates are treated as exact values in subsequent analysis, the software may overlook this additional error. This oversight results in **underestimated standard errors and overly small p-values**, leading to misleading conclusions.

Multiple imputation addresses this issue by generating multiple estimates for each missing value. These estimates differ slightly due to their random component, which reintroduces variation. This variation helps the software incorporate the uncertainty of imputed values, resulting in:

- **Unbiased parameter estimates**
- **Accurate standard errors**
- **Improved p-values**

Multiple imputation was a significant breakthrough in statistics approximately 20 years ago. It provides solutions for many missing data issues (though not all) and, when applied correctly, leads to reliable parameter estimates.

If the proportion of missing data is very small (e.g., 2-3%), the choice of imputation method is less critical.

13.4.4.2 Goals of Multiple Imputation

The primary goals of any missing data technique, including multiple imputation, are:

1. **Unbiased parameter estimates**: Ensuring accurate regression coefficients, group means, odds ratios, etc.
2. **Accurate standard errors**: This leads to reliable p-values and appropriate statistical inferences.
3. **Adequate power**: To detect meaningful and significant parameter values.

13.4.4.3 Overview of Rubin's Framework

Rubin's Framework provides the theoretical foundation for MI. It uses a Bayesian model-based approach for generating imputations and a frequentist approach for evaluating the results. The central goals of Rubin's framework are to ensure that imputations:

- Retain the statistical relationships present in the data.
- Reflect the uncertainty about the true values of the missing data.

Under Rubin's framework, MI offers the following advantages:

- **Generalizability**: Unlike Maximum Likelihood Estimation (MLE), MI can be applied to a wide range of models.
- **Statistical Properties**: When data is MAR or MCAR, MI estimates are consistent, asymptotically normal, and efficient.

Rubin also emphasized the importance of using multiple imputations, as single imputations fail to account for variability in the imputed values, leading to underestimated standard errors and overly optimistic test statistics.

13.4.4.4 Multivariate Imputation via Chained Equations (MICE)

Multivariate Imputation via Chained Equations (MICE) is a widely used algorithm for implementing MI, particularly in datasets with mixed variable types. The steps of MICE include:

1. **Initialization**: Replace missing values with initial guesses, such as the mean or median of the observed data.
2. **Iterative Imputation**:
   - For each variable with missing values, regress it on all other variables (or a subset of relevant predictors).
   - Use the regression model to predict missing values, adding a random error term drawn from the residual distribution.
3. **Convergence**: Repeat the imputation process until parameter estimates stabilize.

MICE offers flexibility in specifying regression models for each variable, accommodating continuous, categorical, and binary data.

13.4.4.5 Bayesian Ridge Regression for Imputation

Bayesian ridge regression is an advanced imputation method that incorporates prior distributions on the regression coefficients, making it particularly useful when:

- Predictors are highly correlated.
- Sample sizes are small.
- Missingness is substantial.

This method treats the regression coefficients as random variables and samples from their posterior distribution, introducing variability into the imputation process. Bayesian ridge regression is more computationally intensive than simpler methods like MICE but offers greater robustness.

13.4.4.6 Combining Results from MI (Rubin's Rules)

Once multiple datasets are imputed and analyzed, Rubin's Rules are used to combine the results. The goal is to properly account for the uncertainty introduced by missing data. For a parameter of interest θ:

1. **Estimate Combination**:

$$\bar{\theta} = \frac{1}{M} \sum_{m=1}^{M} \theta_m$$

   where θ_m is the estimate from the mth imputed dataset, and M is the number of imputations.

2. **Variance Combination**:

$$T = \bar{W} + \left(1 + \frac{1}{M}\right) B$$

   where:

   - $\bar{W}$ is the average within-imputation variance.
   - B is the between-imputation variance:

$$B = \frac{1}{M-1} \sum_{m=1}^{M} (\theta_m - \bar{\theta})^2$$

These formulas adjust the final variance to reflect uncertainty both within and across imputations.

13.4.4.6.1 Challenges

1. **Stochastic Variability**: MI results vary slightly between runs due to its reliance on random draws. To ensure reproducibility, always set a random seed.

2. **Convergence**: Iterative algorithms like MICE may struggle to converge, especially with high proportions of missing data.
3. **Assumption of MAR**: MI assumes that missing data is MAR. If data is Missing Not at Random (MNAR), MI can produce biased results.

13.4.4.6.2 Best Practices

1. **Algorithm Selection**:
   - Use Multiple Imputation by Chained Equations (MICE) for datasets with mixed data types or when relationships between variables are complex.
   - Apply Bayesian Ridge Regression for small datasets or when predictors are highly correlated.
2. **Diagnostic Checks**:
   - Evaluate the quality of imputations and assess convergence using trace plots or diagnostic statistics to ensure reliable results.
3. **Data Transformations**:
   - For skewed or proportion data, consider applying log or logit transformations before imputation and inverse-transforming afterward to preserve the data's original scale.
4. **Handling Non-Linear Relationships**:
   - For non-linear relationships or interactions, stratify imputations by the levels of the categorical variable involved to ensure accurate estimates.
5. **Number of Imputations**:
   - Use at least 20 imputations for small datasets or datasets with high missingness. This ensures robust and reliable results in downstream analyses.
6. **Avoid Rounding Imputations for Dummy Variables**:
   - Many imputation methods (e.g., Markov Chain Monte Carlo [MCMC]) assume normality, even for dummy variables. While it was historically recommended to round imputed values to 0 or 1 for binary variables, research shows that this introduces bias in parameter estimates. Instead, leave imputed values as fractional, even though this may seem counter-intuitive.
7. **Do Not Transform Skewed Variables Before Imputation**:
   - Transforming variables to meet normality assumptions before imputation can distort their relationships with other variables, leading to biased imputations and possibly introducing outliers. It is better to directly impute the skewed variable.
8. **Use More Imputations**:
   - Traditional advice suggests 5–10 imputations are sufficient for unbiased estimates, but inconsistencies may arise in repeated analyses. [@Bodner_2008] suggests using a number of imputations equal to the percentage of missing data. As additional imputations generally do not significantly increase the computational workload, using more

imputations is a prudent choice.

9. **Create Multiplicative Terms Before Imputation**:
   - When your model includes interaction or quadratic terms, generate these terms before imputing missing values. Imputing first and then generating these terms can introduce bias in their regression parameters, as highlighted by (Von Hippel, 2009).

13.5 Evaluation of Imputation Methods

13.5.1 Statistical Metrics for Assessing Imputation Quality

To evaluate the quality of imputed data, several statistical metrics are commonly used. These metrics compare the imputed values to the observed values (in cases where missingness is simulated or artificially introduced) or assess the overall impact of imputation on the quality of subsequent analyses. Key metrics include:

- **Root Mean Squared Error (RMSE):** RMSE is calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

  It measures the average magnitude of errors between the true and imputed values. Lower RMSE indicates better imputation accuracy.

- **Mean Absolute Error (MAE):** MAE measures the average absolute difference between observed and imputed values:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

  MAE provides a straightforward assessment of imputation performance and is less sensitive to outliers than RMSE.

- **Log-Likelihood and Deviance Measures:** Log-likelihood can be used to evaluate how well the imputation model fits the data. Deviance measures, based on likelihood comparisons, assess the relative goodness of fit of imputation models. These are particularly useful in evaluating methods like maximum likelihood estimation.

In practice, these metrics may be combined with graphical methods such as density plots and residual analysis to understand imputation performance more thoroughly.

13.5.2 Bias-Variance Tradeoff in Imputation

Imputation methods must balance bias and variance to achieve reliable results. Simpler methods, such as mean or mode imputation, often lead to biased parameter estimates, particularly if the missingness mechanism is non-random. These methods underestimate variability, shrinking standard errors and potentially leading to overconfidence in statistical inferences.

Conversely, advanced methods like Multiple Imputation or Full Information Maximum Likelihood (FIML) typically yield unbiased estimates with appropriately calibrated variances. However, these methods may increase computational complexity and require careful tuning of assumptions and parameters.

The tradeoff is summarized as follows:

- **High Bias, Low Variance:** Simpler methods (e.g., single imputation, mean imputation).
- **Low Bias, Moderate Variance:** Advanced methods (e.g., MI, FIML, Bayesian methods).

13.5.3 Sensitivity Analysis

Sensitivity analysis is crucial to assess the robustness of imputation methods under varying assumptions. Two primary areas of focus include:

- **Assessing Robustness to Assumptions:** Imputation models often rely on assumptions about the missingness mechanism (See Definition and Classification of Missing Data). Sensitivity analysis involves testing how results vary when these assumptions are slightly relaxed or modified.
- **Impact on Downstream Analysis:** The quality of imputation should also be evaluated based on its influence on downstream analyses (Objectives of Imputation). For instance:
  - Does the imputation affect causal inference in regression models?
  - Are the conclusions from hypothesis testing or predictive modeling robust to the imputation technique?

13.5.4 Validation Using Simulated Data and Real-World Case Studies

Validation of imputation methods is best performed through a combination of simulated data and real-world examples:

1. **Simulated Data:** - Create datasets with known missingness patterns and true values. - Apply various imputation methods and assess their performance using RMSE, MAE, and other metrics.
2. **Real-World Case Studies:**
   - Use datasets from actual studies, such as customer transaction data in marketing or financial data in portfolio analysis.
   - Evaluate the impact of imputation on actionable outcomes (e.g., market segmentation, risk assessment).

Combining these approaches ensures that methods generalize well across different contexts and data structures.

13.6 Criteria for Choosing an Effective Approach

Choosing an appropriate imputation method depends on the following criteria:

1. **Unbiased Parameter Estimates:** The technique should ensure that key estimates, such as means, variances, and regression coefficients, are unbiased, particularly in the presence of MAR or MNAR data.

2. **Adequate Power:** The method should preserve statistical power, enabling robust hypothesis testing and model estimation. This ensures that important effects are not missed due to inflated type II error.

3. **Accurate Standard Errors:** Accurate estimation of standard errors is critical for reliable p-values and confidence intervals. Methods like single imputation often underestimate standard errors, leading to overconfident conclusions.

Preferred Methods: Multiple Imputation and Full Information Maximum Likelihood

Multiple Imputation (MI):

- MI replaces missing values with multiple plausible values drawn from a predictive distribution. It generates multiple complete datasets, analyzes each dataset, and combines the results.

- Pros: Handles uncertainty well, provides valid standard errors, and is robust under MAR.

- Cons: Computationally intensive, sensitive to model mis-specification.

Full Information Maximum Likelihood (FIML):

- FIML uses all available data to estimate parameters directly, avoiding the need to impute missing values explicitly.

- Pros: Efficient, unbiased under MAR, and computationally elegant.
- Cons: Requires correctly specified models and may be sensitive to MNAR data.

Methods to Avoid

- **Single Imputation (e.g., Mean, Mode):**
  - Leads to biased estimates and underestimates variability.
- **Listwise Deletion:**
  - Discards rows with missing data, reducing sample size and potentially introducing bias if the data is not MCAR.

Practical Considerations

- Computational efficiency and ease of implementation.
- Compatibility with downstream analysis methods.
- Alignment with the data's missingness mechanism.

13.7 Challenges and Ethical Considerations

13.7.1 Challenges in High-Dimensional Data

High-dimensional data, where the number of variables exceeds the number of observations, poses unique challenges for missing data analysis.

- **Curse of Dimensionality**: Standard imputation methods, such as mean or regression imputation, struggle with high-dimensional spaces due to sparse data distribution.
- **Regularized Methods**: Techniques such as LASSO, Ridge Regression, and Elastic Net can be used to handle high-dimensional missing data. These methods shrink model coefficients, preventing overfitting.
- **Matrix Factorization**: Methods like Principal Component Analysis (PCA) or Singular Value Decomposition (SVD) are often adapted to impute missing values in high-dimensional datasets by reducing the dimensionality first.

13.7.2 Missing Data in Big Data Contexts

The advent of big data introduces additional complexities for missing data handling, including computational scalability and storage constraints.

13.7.2.1 Distributed Imputation Techniques

- **MapReduce Frameworks**: Algorithms like k-nearest neighbor (KNN) imputation or multiple imputation can be adapted for distributed environ-

ments using MapReduce or similar frameworks.

- **Federated Learning**: In scenarios where data is distributed across multiple locations (e.g., in healthcare or banking), federated learning allows imputation without centralizing data, ensuring privacy.

13.7.2.2 Cloud-Based Implementations

- **Cloud-Native Algorithms**: Cloud platforms like AWS, Google Cloud, and Azure provide scalable solutions for implementing advanced imputation algorithms on large datasets.
- **AutoML Integration**: Automated Machine Learning (AutoML) pipelines often include missing data handling as a preprocessing step, leveraging cloud-based computational power.
- **Real-Time Imputation**: In e-commerce, cloud-based solutions enable real-time imputation for recommendation systems or fraud detection, ensuring seamless user experiences.

13.7.3 Ethical Concerns

13.7.3.1 Bias Amplification

- **Introduction of Systematic Bias**: Imputation methods can inadvertently reinforce existing biases. For example, imputing salary data based on demographic variables may propagate societal inequalities.
- **Business Implications**: In credit scoring, biased imputation of missing financial data can lead to unfair credit decisions, disproportionately affecting marginalized groups.
- **Mitigation Strategies**: Techniques such as fairness-aware machine learning and bias auditing can help identify and reduce bias introduced during imputation.

13.7.3.2 Transparency in Reporting Imputation Decisions

- **Reproducibility and Documentation**: Transparent reporting of imputation methods and assumptions is essential for reproducibility. Analysts should provide clear documentation of the imputation pipeline.
- **Stakeholder Communication**: In business settings, communicating imputation decisions to stakeholders ensures informed decision-making and trust in the results.
- **Ethical Frameworks**: Ethical guidelines, such as those provided by the European Union's GDPR or industry-specific codes, emphasize the importance of transparency in data handling.

13.8 Emerging Trends in Missing Data Handling

13.8.1 Advances in Neural Network Approaches

Neural networks have transformed the landscape of missing data imputation, offering flexible, scalable, and powerful solutions that go beyond traditional methods.

13.8.1.1 Variational Autoencoders (VAEs)

- **Overview**: Variational Autoencoders (VAEs) are generative models that encode data into a latent space and reconstruct it, filling in missing values during reconstruction.
- **Advantages**:
  - Handle complex, non-linear relationships between variables.
  - Scalable to high-dimensional datasets.
  - Generate probabilistic imputations, reflecting uncertainty.
- **Applications**:
  - In marketing, VAEs can impute missing customer behavior data while accounting for seasonal and demographic variations.
  - In finance, VAEs assist in imputing missing stock price data by modeling dependencies among assets.

13.8.1.2 GANs for Missing Data

- **Generative Adversarial Networks (GANs)**: GANs consist of a generator and a discriminator, with the generator imputing missing data and the discriminator evaluating its quality.
- **Advantages**:
  - Preserve data distributions and avoid over-smoothing.
  - Suitable for imputation in datasets with complex patterns or multimodal distributions.
- **Applications**:
  - In healthcare, GANs have been used to impute missing patient records while preserving patient privacy and data integrity.
  - In retail, GANs can model missing sales data to predict trends and optimize inventory.

13.8.2 Integration with Reinforcement Learning

Reinforcement learning (RL) is increasingly being integrated into missing data strategies, particularly in dynamic or sequential data environments.

- **Markov Decision Processes (MDPs)**: RL models missing data handling as an MDP, where actions (imputations) are optimized based on rewards (accuracy of predictions or decisions).
- **Active Imputation**:
  - RL can be used to actively query for missing data points, prioritizing those with the highest impact on downstream tasks.
  - Example: In customer churn prediction, RL can optimize the imputation of high-value customer records.
- **Applications**:
  - Financial forecasting: RL models are used to impute missing transaction data dynamically, optimizing portfolio decisions.
  - Smart cities: RL-based models handle missing sensor data to enhance real-time decision-making in traffic management.

13.8.3 Synthetic Data Generation for Missing Data

Synthetic data generation has emerged as a robust solution to address missing data, providing flexibility and privacy.

- **Data Augmentation**: Synthetic data is generated to augment datasets with missing values, reducing biases introduced by imputation.
- **Techniques**:
  - **Simulations**: Monte Carlo simulations create plausible data points based on observed distributions.
  - **Generative Models**: GANs and VAEs generate realistic synthetic data that aligns with existing patterns.
- **Applications**:
  - In fraud detection, synthetic datasets balance the impact of missing values on anomaly detection.
  - In insurance, synthetic data supports pricing models by filling in gaps from incomplete policyholder records.

13.8.4 Federated Learning and Privacy-Preserving Imputation

Federated learning has gained traction as a method for collaborative analysis while preserving data privacy.

- **Federated Imputation**:
  - Distributed imputation algorithms operate on decentralized data, ensuring that sensitive information remains local.
  - Example: Hospitals collaboratively impute missing patient data without sharing individual records.
- **Privacy Mechanisms**:
  - Differential privacy adds noise to imputed values, protecting individual-level data.
  - Homomorphic encryption allows computations on encrypted data, ensuring privacy throughout the imputation process.
- **Applications**:
  - Healthcare: Federated learning imputes missing diagnostic data across clinics.
  - Banking: Collaborative imputation of financial transaction data supports risk modeling while adhering to regulations.

13.8.5 Imputation in Streaming and Online Data Environments

The increasing use of streaming data in business and technology requires real-time imputation methods to ensure uninterrupted analysis.

- **Challenges**:
  - Imputation must occur dynamically as data streams in.
  - Low latency and high accuracy are essential to maintain real-time decision-making.
- **Techniques**:
  - **Online Learning Algorithms**: Update imputation models incrementally as new data arrives.
  - **Sliding Window Methods**: Use recent data to estimate and impute missing values in real time.
- **Applications**:
  - IoT devices: Imputation in sensor networks for smart homes or industrial monitoring ensures continuous operation despite data transmission issues.
  - Financial markets: Streaming imputation models predict and fill gaps in real-time stock price feeds to inform trading algorithms.

13.9 Application of Imputation

This section demonstrates how to visualize missing data and handle it using different imputation techniques.

Package	Algorithm	Cont Var	Cate Var	Diagnostics	Complexity Handling	Best Use Case	Limitations
missForest	Random Forest	Yes	Yes	Out-of-bag error (NRMSE, PFC)	Handles complex interactions	Mixed data types with complex interactions	May overfit with small datasets
Hmisc	Additive Regression, Bootstrap, Predictive Mean Matching	Yes	Yes	R^2 for imputed values	Basic to intermediate complexity	Simple datasets with low complexity	Limited to simple imputation methods
mi	Bayesian Regression	Yes	Yes	Graphical diagnostics,convergence	Detects issues like collinearity	Datasets with irregularities	Computationally intensive for large data
MICE	Multivariate Imputation via Chained Equations	Yes	Yes	Density plots, pooling of results	Handles variable interactions	General-purpose imputation for MAR data	Requires proper method selection for variable types

Package	Algorithm	Cont Var	Cate Var	Diagnostics	Complexity Handling	Best Use Case	Limitations
Amelia	Bootstrap-based Expectation Maximization (EMB)	Yes	Limited (requires normality)	Diagnostics supported	Works well with large/time series data	Time-series or datasets approximating MVN	Assumes MVN, requires transformations for non-MVN

13.9.1 Visualizing Missing Data

Visualizing missing data is an essential first step in understanding the patterns and extent of missingness in your dataset.

```
library(visdat)
library(naniar)
library(ggplot2)

# Visualizing missing data
vis_miss(airquality)
```

```r
# Missingness patterns using an upset plot
gg_miss_upset(airquality)
```

```r
# Scatter plot of missing data with faceting
ggplot(airquality, aes(x, y)) +
  geom_miss_point() +
  facet_wrap(~ group)

# Missing values by variable
gg_miss_var(data, facet = group)

# Missingness in relation to factors
gg_miss_fct(x = variable1, fct = variable2)
```

For more details, read The Missing Book by Nicholas Tierney & Allison Horst.

13.9.2 How Many Imputations?

Usually, 5 imputations are sufficient unless there is an extremely high proportion of missing data. High proportions require revisiting data collection processes.

Rubin's Rule for Relative Efficiency

According to Rubin, the relative efficiency of an estimate based on m imputations (relative to infinite imputations) is given by:

$$\text{Relative Efficiency} = (1 + \frac{\lambda}{m})^{-1}$$

where λ is the rate of missing data.

For example, with 50% missing data ($\lambda = 0.5$), the standard deviation of an estimate based on 5 imputations is only about 5% wider than that from infinite imputations:

$$\sqrt{1 + \frac{0.5}{5}} = 1.049$$

13.9.3 Generating Missing Data for Demonstration

```
library(missForest)

# Load the data
data <- iris

# Generate 10% missing values at random
set.seed(1)
iris.mis <- prodNA(iris, noNA = 0.1)

# Remove categorical variables for numeric imputation
iris.mis.cat <- iris.mis
iris.mis <- subset(iris.mis, select = -c(Species))
```

13.9.4 Imputation with Mean, Median, and Mode

Mean, median, or mode imputation is a simple yet commonly used technique.

```
# Imputation for the entire dataset
e1071::impute(iris.mis, what = "mean")        # Replace with mean
e1071::impute(iris.mis, what = "median")      # Replace with median

# Imputation by variable
Hmisc::impute(iris.mis$Sepal.Length, mean)    # Replace with mean
Hmisc::impute(iris.mis$Sepal.Length, median)  # Replace with median
Hmisc::impute(iris.mis$Sepal.Length, 0)       # Replace with a specific value
```

Checking Accuracy

Accuracy can be checked by comparing predictions with actual values.

```
# Example data
actuals <- iris$Sepal.Width[is.na(iris.mis$Sepal.Width)]
predicteds <- rep(mean(iris$Sepal.Width, na.rm = TRUE), length(actuals))
```

```r
# Using MLmetrics package
library(MLmetrics)

MAE(predicteds, actuals)
#> [1] 0.2870303
MSE(predicteds, actuals)
#> [1] 0.1301598
RMSE(predicteds, actuals)
#> [1] 0.3607767
```

13.9.5 K-Nearest Neighbors (KNN) Imputation

KNN is a more sophisticated method, leveraging similar observations to fill in missing values.

```r
library(DMwR2)
knnOutput <- knnImputation(data = iris.mis.cat, meth = "median")
anyNA(knnOutput)  # Check for remaining missing values
#> [1] FALSE
```

```r
actuals <- iris$Sepal.Width[is.na(iris.mis$Sepal.Width)]
predicteds <- knnOutput[is.na(iris.mis$Sepal.Width), "Sepal.Width"]
# Using MLmetrics package
library(MLmetrics)

MAE(predicteds, actuals)
#> [1] 0.2318182
MSE(predicteds, actuals)
#> [1] 0.1038636
RMSE(predicteds, actuals)
#> [1] 0.3222788
```

KNN typically improves upon mean or median imputation in terms of predictive accuracy.

13.9.6 Imputation with Decision Trees (rpart)

Decision trees, such as those implemented in `rpart`, are effective for both numeric and categorical variables.

```r
library(rpart)

# Imputation for a categorical variable
class_mod <- rpart(
  Species ~ . - Sepal.Length,
  data = iris.mis.cat[!is.na(iris.mis.cat$Species), ],
```

```
  method = "class",
  na.action = na.omit
)

# Imputation for a numeric variable
anova_mod <- rpart(
  Sepal.Width ~ . - Sepal.Length,
  data = iris.mis[!is.na(iris.mis$Sepal.Width), ],
  method = "anova",
  na.action = na.omit
)

# Predictions
species_pred <- predict(class_mod, iris.mis.cat[is.na(iris.mis.cat$Species), ])
width_pred <- predict(anova_mod, iris.mis[is.na(iris.mis$Sepal.Width), ])
```

13.9.7 MICE (Multivariate Imputation via Chained Equations)

MICE assumes that the data are **Missing at Random (MAR)**. It imputes data for each variable by specifying an imputation model tailored to the variable type.

13.9.7.1 How MICE Works

For a dataset with variables $X_1, X_2, \dots, X_k$:

- If X_1 has missing data, it is regressed on the other variables.
- This process is repeated for all variables with missing data, using the previously predicted values as needed.

By default:

- **Continuous variables** use linear regression.
- **Categorical variables** use logistic regression.

13.9.7.2 Methods Available in MICE

- `pmm` **(Predictive Mean Matching)**: For numeric variables.
- `logreg` **(Logistic Regression)**: For binary variables (2 levels).
- `polyreg` **(Bayesian polytomous regression)**: For factor variables (2 levels).
- **Proportional Odds Model**: For ordered factor variables (2 levels).

```
# Load packages
library(mice)
```

```r
library(VIM)

# Check missing values pattern
md.pattern(iris.mis)
```

```
#>     Sepal.Width Sepal.Length Petal.Length Petal.Width
#> 100           1            1            1           1  0
#> 15            1            1            1           0  1
#> 8             1            1            0           1  1
#> 2             1            1            0           0  2
#> 11            1            0            1           1  1
#> 1             1            0            1           0  2
#> 1             1            0            0           1  2
#> 1             1            0            0           0  3
#> 7             0            1            1           1  1
#> 3             0            1            0           1  2
#> 1             0            0            1           1  2
#>              11           15           15          19 60
```

```r
# Plot missing values
aggr(
  iris.mis,
  col = c('navyblue', 'yellow'),
  numbers = TRUE,
  sortVars = TRUE,
  labels = names(iris.mis),
  cex.axis = 0.7,
```

```
  gap = 3,
  ylab = c("Missing data", "Pattern")
)
```

```
#>
#>  Variables sorted by number of missings:
#>      Variable      Count
#>   Petal.Width 0.12666667
#>  Sepal.Length 0.10000000
#>  Petal.Length 0.10000000
#>   Sepal.Width 0.07333333
```

Imputing Data

```r
# Perform multiple imputation using MICE
imputed_Data <- mice(
  iris.mis,
  m = 5,             # Number of imputed datasets
  maxit = 10,        # Number of iterations
  method = 'pmm',    # Imputation method
  seed = 500         # Random seed for reproducibility
)
```

Evaluating Imputed Data

```r
# Summary of imputed data
summary(imputed_Data)
#> Class: mids
#> Number of multiple imputations:  5
```

```
#> Imputation methods:
#> Sepal.Length  Sepal.Width Petal.Length  Petal.Width
#>        "pmm"        "pmm"        "pmm"        "pmm"
#> PredictorMatrix:
#>              Sepal.Length Sepal.Width Petal.Length Petal.Width
#> Sepal.Length            0           1            1           1
#> Sepal.Width             1           0            1           1
#> Petal.Length            1           1            0           1
#> Petal.Width             1           1            1           0

# Density plot: compare imputed values (red) with observed values (blue)
densityplot(imputed_Data)
```

Accessing and Using Imputed Data

```
# Access the complete datasets
completeData1 <- complete(imputed_Data, 1)  # First imputed dataset
completeData2 <- complete(imputed_Data, 2)  # Second imputed dataset
```

Regression Model with Imputed Dataset

```
# Fit a regression model using imputed datasets
fit <- with(data = imputed_Data, exp = lm(Sepal.Width ~ Sepal.Length + Petal.Width))

# Combine results of all 5 models
combine <- pool(fit)
summary(combine)
#>           term   estimate  std.error statistic        df      p.value
```

```
#> 1  (Intercept)  1.9054698 0.33454626  5.695684 105.12438 1.127064e-07
#> 2 Sepal.Length  0.2936285 0.07011405  4.187870  88.69066 6.625536e-05
#> 3  Petal.Width -0.4742921 0.08138313 -5.827892  46.94941 4.915270e-07
```

13.9.8 Amelia

Amelia uses a **bootstrap-based Expectation-Maximization with Bootstrapping (EMB) algorithm** for imputation, making it faster and suitable for cross-sectional and time-series data.

13.9.8.1 Assumptions

- All variables must follow a **Multivariate Normal Distribution (MVN)**. Transformations may be required for non-normal data.
- Data must be **Missing at Random (MAR)**.

13.9.8.2 Comparison: Amelia vs. MICE

- **MICE** imputes on a variable-by-variable basis using separate models.
- **Amelia** uses a joint modeling approach based on MVN.
- **MICE** handles multiple data types, while **Amelia** requires variables to approximate normality.

13.9.8.3 Imputation with Amelia

```r
library(Amelia)
data("iris")

# Seed 10% missing values
set.seed(123)
iris.mis <- prodNA(iris, noNA = 0.1)

# Specify columns and run Amelia
amelia_fit <- amelia(
  iris.mis,
  m = 5,                        # Number of imputations
  parallel = "multicore",       # Use multicore processing
  noms = "Species"              # Nominal variables
)
#> -- Imputation 1 --
#>
#>   1  2  3  4  5  6  7
#>
#> -- Imputation 2 --
```

```
#>
#>   1  2  3  4  5
#>
#> -- Imputation 3 --
#>
#>   1  2  3  4  5
#>
#> -- Imputation 4 --
#>
#>   1  2  3  4  5  6
#>
#> -- Imputation 5 --
#>
#>   1  2  3  4  5  6  7  8  9 10

# Access imputed outputs
# amelia_fit$imputations[[1]]
```

Amelia's workflow includes bootstrapping multiple imputations to generate robust estimates of means and variances. This process ensures flexibility and speed for large datasets.

13.9.9 missForest

The `missForest` package provides a robust non-parametric imputation method using the Random Forest algorithm. It is versatile, handling both continuous and categorical variables without requiring assumptions about the underlying functional forms.

Key Features of missForest

1. **Non-Parametric**: No assumptions about the functional form.
2. **Variable-Specific Models**: Builds a random forest model for each variable to impute missing values.
3. **Error Estimates**: Provides out-of-bag (OOB) imputation error estimates.
   - **NRMSE** (Normalized Root Mean Squared Error) for continuous variables.
   - **PFC** (Proportion of Falsely Classified) for categorical variables.
4. **High Control**: Offers customizable parameters like `mtry` and `ntree`.

```r
library(missForest)

# Impute missing values using default parameters
iris.imp <- missForest(iris.mis)

# Check imputed values
```

```r
# View the imputed dataset
# iris.imp$ximp
```

```r
# Out-of-bag error estimates
iris.imp$OOBerror
#>      NRMSE        PFC
#> 0.14004144 0.02877698

# Compare imputed data with original data to calculate error
iris.err <- mixError(iris.imp$ximp, iris.mis, iris)
iris.err
#>      NRMSE        PFC
#> 0.14420833 0.09090909
```

13.9.10 Hmisc

The `Hmisc` package provides a suite of tools for imputing missing data, offering both simple methods (like mean or median imputation) and more advanced approaches like `aregImpute`.

Features of Hmisc

1. **`impute()`**: Simple imputation using user-defined methods like mean, median, or a random value.
2. **`aregImpute()`**:
   - Combines additive regression, bootstrapping, and predictive mean matching.
   - Handles continuous and categorical variables.
   - Automatically recognizes variable types and applies appropriate methods.

Assumptions

- Linearity in the variables being predicted.
- Fisher's optimum scoring is used for categorical variable prediction.

```r
library(Hmisc)

# Impute using mean
iris.mis$imputed_SepalLength <- with(iris.mis, impute(Sepal.Length, mean))

# Impute using random value
iris.mis$imputed_SepalLength2 <- with(iris.mis, impute(Sepal.Length, 'random'))

# Advanced imputation using aregImpute
```

```
impute_arg <- aregImpute(
  ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width + Species,
  data = iris.mis,
  n.impute = 5
)
#> Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8

# Check R-squared values for predicted missing values
impute_arg
#>
#> Multiple Imputation using Bootstrap and PMM
#>
#> aregImpute(formula = ~Sepal.Length + Sepal.Width + Petal.Length +
#>     Petal.Width + Species, data = iris.mis, n.impute = 5)
#>
#> n: 150   p: 5    Imputations: 5      nk: 3
#>
#> Number of NAs:
#> Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
#>           17           19           12           16           11
#>
#>              type d.f.
#> Sepal.Length    s    2
#> Sepal.Width     s    2
#> Petal.Length    s    2
#> Petal.Width     s    2
#> Species         c    2
#>
#> Transformation of Target Variables Forced to be Linear
#>
#> R-squares for Predicting Non-Missing Values for Each Variable
#> Using Last Imputations of Predictors
#> Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
#>        0.895        0.536        0.987        0.967        0.984

# Access imputed values for Sepal.Length
impute_arg$imputed$Sepal.Length
#>     [,1] [,2] [,3] [,4] [,5]
```

```
#> 13   4.4  4.9  4.9  5.0  4.9
#> 14   4.8  4.4  5.0  4.5  4.5
#> 23   4.8  5.1  5.1  5.1  4.8
#> 26   5.0  4.8  4.9  4.9  5.0
#> 34   5.0  5.8  6.0  5.7  5.8
#> 39   4.4  4.9  5.0  4.5  4.6
#> 41   5.2  5.1  4.8  5.0  4.8
#> 69   5.8  6.0  6.3  6.0  6.1
#> 72   5.6  5.7  5.7  5.8  6.1
#> 89   6.1  5.7  5.7  5.6  6.9
#> 90   5.5  6.2  5.2  6.0  5.8
#> 91   5.7  6.9  6.0  6.4  6.4
#> 116  5.9  6.8  6.4  6.6  6.9
#> 118  7.9  7.9  7.9  7.9  7.9
#> 135  6.7  6.7  6.7  6.9  6.7
#> 141  7.0  6.3  5.9  6.7  7.0
#> 143  5.7  6.7  5.8  6.3  5.4
```

Note: While `missForest` often outperforms `Hmisc` in terms of accuracy, the latter is useful for datasets with simpler requirements.

13.9.11 mi

The `mi` package is a powerful tool for imputation, using Bayesian methods and providing rich diagnostics for model evaluation and convergence.

Features of mi

1. **Graphical Diagnostics**: Visualize imputation models and convergence.
2. **Bayesian Regression**: Handles separation and other issues in data.
3. **Irregularity Detection**: Automatically detects issues like high collinearity.
4. **Noise Addition**: Adds noise to address additive constraints.

```r
library(mi)

# Perform imputation using mi
mi_data <- mi(iris.mis, seed = 1)

# Summary of the imputation process
summary(mi_data)
#> $Sepal.Length
#> $Sepal.Length$is_missing
#> missing
#> FALSE  TRUE
```

```
#>   133    17
#>
#> $Sepal.Length$imputed
#>       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
#> -0.3997862 -0.0207533 -0.0000533  0.0237651  0.0320814  0.7490046
#>
#> $Sepal.Length$observed
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> -0.90110 -0.47329 -0.04549  0.00000  0.32120  1.23792
#>
#>
#> $Sepal.Width
#> $Sepal.Width$is_missing
#> missing
#> FALSE  TRUE
#>   131    19
#>
#> $Sepal.Width$imputed
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> -1.81801 -0.50915 -0.05826 -0.07340  0.36536  1.42910
#>
#> $Sepal.Width$observed
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> -1.01272 -0.30642 -0.07099  0.00000  0.39988  1.34161
#>
#>
#> $Petal.Length
#> $Petal.Length$is_missing
#> missing
#> FALSE  TRUE
#>   138    12
#>
#> $Petal.Length$imputed
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> -0.88552 -0.59436  0.22812  0.05184  0.51312  0.85922
#>
#> $Petal.Length$observed
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> -0.7797 -0.6088  0.1459  0.0000  0.3880  0.9006
#>
#>
#> $Petal.Width
#> $Petal.Width$is_missing
#> missing
#> FALSE  TRUE
```

```
#>   134    16
#>
#> $Petal.Width$imputed
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> -1.01726 -0.09527  0.22662  0.15893  0.55941  0.85487
#>
#> $Petal.Width$observed
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> -0.69624 -0.56602  0.08503  0.00000  0.41055  0.86629
#>
#>
#> $Species
#> $Species$crosstab
#>
#>              observed imputed
#>   setosa          180      17
#>   versicolor      192      12
#>   virginica       184      15
#>
#>
#> $imputed_SepalLength
#> $imputed_SepalLength$is_missing
#> [1] "all values observed"
#>
#> $imputed_SepalLength$observed
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> -0.9574 -0.4379  0.0000  0.0000  0.3413  1.3152
#>
#>
#> $imputed_SepalLength2
#> $imputed_SepalLength2$is_missing
#> [1] "all values observed"
#>
#> $imputed_SepalLength2$observed
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> -0.90570 -0.48398 -0.06225  0.00000  0.35947  1.20292
```

Chapter 14

Model Specification Tests

Model specification tests are critical in econometric analysis to verify whether the assumptions underlying a model hold true. These tests help determine if the model is correctly specified, ensuring that the estimators are both reliable and efficient. A mis-specified model can lead to biased, inconsistent, or inefficient estimates, which undermines the validity of inferences drawn from the analysis.

This chapter addresses various model specification tests, including tests for:

- Nested Model Tests
- Non-Nested Model Tests
- Heteroskedasticity Tests
- Functional Form Tests
- Autocorrelation Tests
- Multicollinearity Diagnostics

Understanding these tests allows researchers to evaluate the robustness of their models and make necessary adjustments to improve model performance.

14.1 Nested Model Tests

Nested models are those where the **restricted model** is a special case of the **unrestricted model**. In other words, the restricted model can be derived from the unrestricted model by imposing constraints on certain parameters, typically setting them equal to zero. This structure allows us to formally test whether the additional variables in the unrestricted model significantly improve the model's explanatory power. The following tests help compare these models:

- Wald Test: Assesses the significance of individual coefficients or groups of coefficients.

- Likelihood Ratio Test: Compares the goodness-of-fit between restricted and unrestricted models.
- F-Test: Evaluates the joint significance of multiple coefficients.
- Chow Test: Evaluates whether the coefficients of a regression model are the same across different groups or time periods.

Consider the following models:

$$\begin{aligned} \textbf{Unrestricted Model:} \quad & y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon \\ \textbf{Restricted Model:} \quad & y = \beta_0 + \beta_1 x_1 + \epsilon \end{aligned}$$

- The **unrestricted model** includes all potential explanatory variables: x_1, x_2, and x_3.
- The **restricted model** is nested within the unrestricted model, containing a subset of variables (in this case, excluding x_2 and x_3).

Our goal is to test the null hypothesis that the restrictions are valid:

$$H_0 : \beta_2 = \beta_3 = 0 \quad \text{(restrictions are valid)}$$

against the alternative hypothesis:

$$H_1 : \text{At least one of } \beta_2, \beta_3 \neq 0 \quad \text{(restrictions are invalid)}$$

To conduct this test, we use the following methods:

14.1.1 Wald Test

The **Wald Test** assesses whether certain linear restrictions on the parameters of the model are valid. It is commonly used when testing the joint significance of multiple coefficients.

Consider a set of linear restrictions expressed as:

$$H_0 : R\beta = r$$

where:

- R is a $q \times k$ restriction matrix,
- β is the $k \times 1$ vector of parameters,
- r is a $q \times 1$ vector representing the hypothesized values (often zeros),
- q is the number of restrictions being tested.

For example, if we want to test $H_0 : \beta_2 = \beta_3 = 0$, then:

$$R = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad r = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The Wald statistic is calculated as:

$$W = (R\hat{\beta} - r)' \left[R \, \hat{\mathrm{Var}}(\hat{\beta}) \, R' \right]^{-1} (R\hat{\beta} - r)$$

Where:

- $\hat{\beta}$ is the vector of estimated coefficients from the unrestricted model,
- $\hat{\mathrm{Var}}(\hat{\beta})$ is the estimated covariance matrix of $\hat{\beta}$.

Distribution and Decision Rule

- Under H_0, the Wald statistic follows a χ^2 distribution with q degrees of freedom:

$$W \sim \chi^2_q$$

- **Decision Rule:**
  - Reject H_0 if $W > \chi^2_{q,\alpha}$, where α is the significance level.
  - A large Wald statistic indicates that the restrictions are invalid.

14.1.2 Likelihood Ratio Test

The **Likelihood Ratio Test** compares the goodness-of-fit between the restricted and unrestricted models. It evaluates whether the additional parameters in the unrestricted model significantly improve the likelihood of observing the data.

Same as before:

$$H_0 : \beta_2 = \beta_3 = 0 \quad \text{vs.} \quad H_1 : \text{At least one of } \beta_2, \beta_3 \neq 0$$

The LR statistic is calculated as:

$$LR = -2 \left(\ln L_R - \ln L_U \right)$$

Where:

- L_R is the maximized likelihood of the restricted model,
- L_U is the maximized likelihood of the unrestricted model.

Distribution and Decision Rule

- Under H_0, the LR statistic follows a χ^2 distribution with q degrees of freedom:

$$LR \sim \chi^2_q$$

- **Decision Rule:**
  - Reject H_0 if $LR > \chi^2_{q,\alpha}$.
  - A large LR statistic suggests that the unrestricted model provides a significantly better fit.

Connection to OLS

In the case of linear regression with normally distributed errors, the LR statistic can be expressed in terms of the sum of squared residuals (SSR):

$$LR = n \ln \left(\frac{SSR_R}{SSR_U} \right)$$

where n is the sample size.

14.1.3 F-Test (for Linear Regression)

The **F-Test** is commonly used in linear regression to evaluate the joint significance of multiple coefficients. It compares the fit of the restricted and unrestricted models using their sum of squared residuals.

Again:

$$H_0 : \beta_2 = \beta_3 = 0 \quad \text{vs.} \quad H_1 : \text{At least one of } \beta_2, \beta_3 \neq 0$$

The F-statistic is calculated as:

$$F = \frac{(SSR_R - SSR_U)/q}{SSR_U/(n-k)}$$

Where:

- SSR_R = Sum of Squared Residuals from the restricted model,
- SSR_U = Sum of Squared Residuals from the unrestricted model,
- q = Number of restrictions (here, 2),
- n = Sample size,
- k = Number of parameters in the unrestricted model.

Distribution and Decision Rule

- Under H_0, the F-statistic follows an F-distribution with $(q, n-k)$ degrees of freedom:

$$F \sim F_{q,n-k}$$

- **Decision Rule:**
  - Reject H_0 if $F > F_{q,n-k,\alpha}$.
  - A large F-statistic indicates that the restricted model fits significantly worse, suggesting the excluded variables are important.

14.1.4 Chow Test

The **Chow Test** evaluates whether the coefficients of a regression model are the same across different groups or time periods. It is often used to detect **structural breaks** in the data.

Key Question:
Should we run two different regressions for two groups, or can we pool the data and use a single regression?

Chow Test Procedure

1. Estimate the regression model for the **pooled data** (all observations).
2. Estimate the model separately for **Group 1** and **Group 2**.
3. Compare the **sum of squared residuals (SSR)** from these models.

The test statistic follows an F-distribution:

$$F = \frac{(SSR_P - (SSR_1 + SSR_2))/k}{(SSR_1 + SSR_2)/(n_1 + n_2 - 2k)}$$

Where:

- SSR_P = Sum of Squared Residuals for the pooled model
- SSR_1, SSR_2 = SSRs for Group 1 and Group 2 models
- k = Number of parameters
- n_1, n_2 = Number of observations in each group

Interpretation:

- A **significant F-statistic** suggests structural differences between groups, implying separate regressions are more appropriate.
- A **non-significant F-statistic** indicates no structural break, supporting the use of a pooled model.

```r
# Load necessary libraries
library(car)        # For Wald Test
library(lmtest)     # For Likelihood Ratio Test
```

```r
library(strucchange)  # For Chow Test

# Simulated dataset
set.seed(123)
n <- 100
x1 <- rnorm(n)
x2 <- rnorm(n)
x3 <- rnorm(n)
epsilon <- rnorm(n)
y <- 2 + 1.5 * x1 + 0.5 * x2 - 0.7 * x3 + epsilon

# Creating a group variable (simulating a structural break)
group <- rep(c(0, 1), each = n / 2)  # Group 0 and Group 1

# ----------------------------------------------------------------------
# Wald Test
# ----------------------------------------------------------------------
unrestricted_model <- lm(y ~ x1 + x2 + x3)    # Unrestricted model
restricted_model <- lm(y ~ x1)                # Restricted model

wald_test <- linearHypothesis(unrestricted_model, c("x2 = 0", "x3 = 0"))
print(wald_test)
#> Linear hypothesis test
#>
#> Hypothesis:
#> x2 = 0
#> x3 = 0
#>
#> Model 1: restricted model
#> Model 2: y ~ x1 + x2 + x3
#>
#>   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
#> 1     98 182.26
#> 2     96 106.14  2    76.117 34.421 5.368e-12 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# ----------------------------------------------------------------------
# Likelihood Ratio Test
# ----------------------------------------------------------------------
lr_test <- lrtest(unrestricted_model, restricted_model)
print(lr_test)
#> Likelihood ratio test
#>
#> Model 1: y ~ x1 + x2 + x3
```

```
#> Model 2: y ~ x1
#>   #Df  LogLik Df  Chisq Pr(>Chisq)
#> 1   5 -144.88
#> 2   3 -171.91 -2 54.064  1.821e-12 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# ----------------------------------------------------------------------
# F-Test (for Linear Regression)
# ----------------------------------------------------------------------
SSR_U <- sum(residuals(unrestricted_model)^2)  # SSR for unrestricted model
SSR_R <- sum(residuals(restricted_model)^2)    # SSR for restricted model
q <- 2                                        # Number of restrictions
n <- length(y)                                # Sample size
k <- length(coef(unrestricted_model))         # Number of parameters in unrestricted model

# F-statistic formula
F_statistic <- ((SSR_R - SSR_U) / q) / (SSR_U / (n - k))
p_value_F <- pf(F_statistic, df1 = q, df2 = n - k, lower.tail = FALSE)

cat("F-statistic:", F_statistic, "\n")
#> F-statistic: 34.42083
cat("P-value:", p_value_F, "\n")
#> P-value: 5.367912e-12

# ----------------------------------------------------------------------
# Chow Test (Proper Use of the Group Variable)
# ----------------------------------------------------------------------
# Pooled model (all data)
pooled_model <- lm(y ~ x1 + x2 + x3)

# Separate models for Group 0 and Group 1
model_group0 <- lm(y[group == 0] ~ x1[group == 0] + x2[group == 0] + x3[group == 0])
model_group1 <- lm(y[group == 1] ~ x1[group == 1] + x2[group == 1] + x3[group == 1])

# Calculating SSRs
SSR_pooled <- sum(residuals(pooled_model)^2)
SSR_group0 <- sum(residuals(model_group0)^2)
SSR_group1 <- sum(residuals(model_group1)^2)

# Chow Test formula
k_chow <- length(coef(pooled_model))  # Number of parameters (including intercept)
n0 <- sum(group == 0)                 # Sample size for Group 0
n1 <- sum(group == 1)                 # Sample size for Group 1
```

```r
F_chow <- ((SSR_pooled - (SSR_group0 + SSR_group1)) / k_chow) /
          ((SSR_group0 + SSR_group1) / (n0 + n1 - 2 * k_chow))

# Corresponding p-value
p_value_chow <-
    pf(
        F_chow,
        df1 = k_chow,
        df2 = (n0 + n1 - 2 * k_chow),
        lower.tail = FALSE
    )

cat("Chow Test F-statistic:", F_chow, "\n")
#> Chow Test F-statistic: 0.3551197
cat("P-value:", p_value_chow, "\n")
#> P-value: 0.8398657
```

Interpretation of the Results

1. **Wald Test**
   - **Null Hypothesis** (H_0): $\beta_2 = \beta_3 = 0$ (the coefficients for x_2 and x_3 are zero).
   - **Decision Rule:**
     - **Reject** H_0 if *p-value* < 0.05: x_2 and x_3 are jointly significant.
     - **Fail to reject** H_0 if *p-value* $\quad 0.05$: x_2 and x_3 do not significantly contribute to the model.
2. **Likelihood Ratio Test (LR Test)**
   - **Null Hypothesis** (H_0): The restricted model fits the data as well as the unrestricted model.
   - **Decision Rule:**
     - **Reject** H_0 if *p-value* < 0.05: The unrestricted model fits better, indicating x_2 and x_3 improve the model.
     - **Fail to reject** H_0 if *p-value* $\quad 0.05$: Adding x_2 and x_3 doesn't improve the model significantly.
3. **F-Test (for Linear Regression)**
   - **Null Hypothesis** (H_0): $\beta_2 = \beta_3 = 0$ (similar to the Wald Test).
   - **Decision Rule:**
     - **Reject** H_0 if *p-value* < 0.05: The excluded variables are significant.

  - **Fail to reject** H_0 if *p-value* 0.05: The excluded variables are not significant.

4. **Chow Test (Using the `group` Variable)**
   - **Null Hypothesis** (H_0): No structural break exists; the regression coefficients are the same across **Group 0** and **Group 1**.
   - **Decision Rule:**
     - **Reject** H_0 if *p-value* < 0.05: A structural break exists—model coefficients differ significantly between the groups.
     - **Fail to reject** H_0 if *p-value* 0.05: No significant structural break detected; the model coefficients are stable across both groups.

14.2 Non-Nested Model Tests

Comparing models is essential to identify which specification best explains the data. While nested model comparisons rely on one model being a restricted version of another, **non-nested models** do not share such a hierarchical structure. This situation commonly arises when comparing models with:

- Different functional forms (e.g., linear vs. logarithmic relationships),
- Different sets of explanatory variables,
- Competing theoretical frameworks.

To compare non-nested models, we rely on specialized statistical tests designed to handle these complexities. The two most widely used approaches are:

- Vuong Test used to compare the fit of two non-nested models to determine which model better explains the data.
- Davidson–MacKinnon J-Test is a regression-based approach for comparing non-nested models. It evaluates which model better fits the data by incorporating the predicted values from the competing model as an additional regressor.

Consider two competing models:

- **Model A:**

$$y = \alpha_0 + \alpha_1 f(X) + \epsilon_A$$

- **Model B:**

$$y = \beta_0 + \beta_1 g(Z) + \epsilon_B$$

Where:

- $f(X)$ and $g(Z)$ represent different functional forms or different sets of explanatory variables.
- The models are **non-nested** because one cannot be obtained from the other by restricting parameters.

Our goal is to determine which model better explains the data.

14.2.1 Vuong Test

The **Vuong Test** is a likelihood-ratio-based approach for comparing non-nested models (Vuong, 1989). It is particularly useful when both models are estimated via Maximum Likelihood Estimation.

Hypotheses

- Null Hypothesis (H_0): Both models are equally close to the true data-generating process (i.e., the models have equal predictive power).
- Alternative Hypothesis (H_1):
  - Positive Test Statistic ($V > 0$): Model A is preferred.
  - Negative Test Statistic ($V < 0$): Model B is preferred.

Vuong Test Statistic

The Vuong test is based on the difference in the log-likelihood contributions of each observation under the two models. Let:

- $\ell_{A,i}$ = log-likelihood of observation i under Model A,
- $\ell_{B,i}$ = log-likelihood of observation i under Model B.

Define the difference in log-likelihoods:

$$m_i = \ell_{A,i} - \ell_{B,i}$$

The Vuong test statistic is:

$$V = \frac{\sqrt{n}\,\bar{m}}{s_m}$$

Where:

- $\bar{m} = \frac{1}{n}\sum_{i=1}^{n} m_i$ is the sample mean of the log-likelihood differences,
- $s_m = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(m_i - \bar{m})^2}$ is the sample standard deviation of m_i,

- n is the sample size.

Distribution and Decision Rule

- Under H_0, the Vuong statistic asymptotically follows a **standard normal distribution**:

$$V \sim N(0, 1)$$

- **Decision Rule:**
  - If $|V| > z_{\alpha/2}$ (critical value from the standard normal distribution), **reject** H_0.
    - If $V > 0$: Prefer **Model A**.
    - If $V < 0$: Prefer **Model B**.
  - If $|V| \leq z_{\alpha/2}$: **Fail to reject** H_0; no significant difference between models.

Corrections for Model Complexity

When comparing models with different numbers of parameters, a **penalized version** of the Vuong test can be used, similar to adjusting for model complexity in criteria like AIC or BIC. The corrected statistic is:

$$V_{\text{adjusted}} = V - \frac{(k_A - k_B)\ln(n)}{2s_m\sqrt{n}}$$

Where k_A and k_B are the number of parameters in Models A and B, respectively.

Limitations of the Vuong Test

- Requires models to be estimated via Maximum Likelihood.
- Sensitive to **model misspecification** and **heteroskedasticity**.
- Assumes **independent and identically distributed (i.i.d.)** errors.

14.2.2 Davidson–MacKinnon J-Test

The **Davidson–MacKinnon J-Test** provides a flexible, regression-based approach for comparing non-nested models (Davidson and MacKinnon, 1981). It evaluates whether the predictions from one model contain information not captured by the competing model. This test can be thought of as comparing models with transformed independent variables, as opposed to the next section, Comparing Models with Transformed Dependent Variables.

Hypotheses

- Null Hypothesis (H_0): The competing model does not provide additional explanatory power beyond the current model.
- Alternative Hypothesis (H_1): The competing model provides additional explanatory power.

Procedure

Consider two competing models:

- **Model A:**

$$y = \alpha_0 + \alpha_1 x + \epsilon_A$$

- **Model B:**

$$y = \beta_0 + \beta_1 \ln(x) + \epsilon_B$$

Step 1: Testing Model A Against Model B

1. **Estimate Model B** and obtain predicted values $\hat{y}_B$.
2. **Run the auxiliary regression:**

$$y = \alpha_0 + \alpha_1 x + \gamma \hat{y}_B + u$$

3. **Test the null hypothesis:**

$$H_0 : \gamma = 0$$

- If γ is significant, Model B adds explanatory power beyond Model A.
- If γ is not significant, Model A sufficiently explains the data.

Step 2: Testing Model B Against Model A

1. **Estimate Model A** and obtain predicted values $\hat{y}_A$.
2. **Run the auxiliary regression:**

$$y = \beta_0 + \beta_1 \ln(x) + \gamma \hat{y}_A + u$$

3. **Test the null hypothesis:**

$$H_0 : \gamma = 0$$

Decision Rules

- **Reject** H_0 in Step 1, Fail to Reject in Step 2: Prefer **Model B**.
- **Fail to Reject** H_0 in Step 1, Reject in Step 2: Prefer **Model A**.
- **Reject** H_0 in Both Steps: Neither model is adequate; reconsider the functional form.
- **Fail to Reject** H_0 in Both Steps: No strong evidence to prefer one model; rely on other criteria (e.g., theory, simplicity). Alternatively, R^2_{adj} can also be used to choose between the two.

Adjusted R^2

- R^2 will always increase with more variables included
- Adjusted R^2 tries to correct by penalizing inclusion of unnecessary variables.

$$\begin{aligned} R^2 &= 1 - \frac{SSR/n}{SST/n} \\ R^2_{adj} &= 1 - \frac{SSR/(n-k)}{SST/(n-1)} \\ &= 1 - \frac{(n-1)(1-R^2)}{(n-k)} \end{aligned}$$

- R^2_{adj} increases if and only if the t-statistic on the additional variable is greater than 1 in absolute value.
- R^2_{adj} is valid in models where there is no heteroskedasticity
- there fore it **should not** be used in determining which variables should be included in the model (the t or F-tests are more appropriate)

14.2.3 Adjusted R^2

The **coefficient of determination** (R^2) measures the proportion of the variance in the dependent variable that is explained by the model. However, a key limitation of R^2 is that it **always increases** (or at least stays the same) when additional explanatory variables are added to the model, even if those variables are not statistically significant.

To address this issue, the **adjusted** R^2 introduces a penalty for including unnecessary variables, making it a more reliable measure when comparing models with different numbers of predictors.

Formulas

Unadjusted R^2:

$$R^2 = 1 - \frac{SSR}{SST}$$

Where:

- SSR = **Sum of Squared Residuals** (measures unexplained variance),
- SST = **Total Sum of Squares** (measures total variance in the dependent variable).

Adjusted R^2:

$$R^2_{\text{adj}} = 1 - \frac{SSR/(n-k)}{SST/(n-1)}$$

Alternatively, it can be expressed in terms of R^2 as:

$$R^2_{\text{adj}} = 1 - \left(\frac{(1-R^2)(n-1)}{n-k}\right)$$

Where:

- n = Number of observations,
- k = Number of estimated parameters in the model (including the intercept).

Key Insights

- **Penalty for Complexity:** Unlike R^2, the adjusted R^2 **can decrease** when irrelevant variables are added to the model because it adjusts for the number of predictors relative to the sample size.
- **Interpretation:** It represents the proportion of variance explained **after accounting for model complexity**.
- **Comparison Across Models:** Adjusted R^2 is useful for comparing models with different numbers of predictors, as it discourages overfitting.

When Does Adjusted R^2 Increase?

- The adjusted R^2 will **increase** if and only if the inclusion of a new variable **improves the model more than expected by chance**.
- **Mathematically:** This typically occurs when the **absolute value of the** t-statistic for the new variable is **greater than 1** (assuming large samples and standard model assumptions).

Limitations of Adjusted R^2

- **Sensitive to Assumptions:** Adjusted R^2 assumes **homoskedasticity** (constant variance of errors) and **no autocorrelation**. In the presence of heteroskedasticity, its interpretation may be misleading.

- **Not a Substitute for Hypothesis Testing:** It **should not** be the primary criterion for deciding which variables to include in a model.
  - Use t-tests to evaluate the significance of individual coefficients.
  - Use F-tests for assessing the joint significance of multiple variables.

14.2.4 Comparing Models with Transformed Dependent Variables

When comparing regression models with different transformations of the dependent variable, such as level and log-linear models, direct comparisons using traditional goodness-of-fit metrics like R^2 or adjusted R^2 are invalid. This is because the transformation changes the scale of the dependent variable, affecting the calculation of the Total Sum of Squares (SST), which is the denominator in R^2 calculations.

Model Specifications

1. **Level Model (Linear):**

$$y = \beta_0 + \beta_1 x_1 + \epsilon$$

2. **Log-Linear Model:**

$$\ln(y) = \beta_0 + \beta_1 x_1 + \epsilon$$

Where:

- y is the dependent variable,
- x_1 is an independent variable,
- ϵ represents the error term.

Interpretation of Coefficients

- **In the Level Model:**
  The effect of x_1 on y is **constant**, regardless of the magnitude of y. Specifically, a **one-unit increase** in x_1 results in a change of β_1 units in y. This implies:

$$\Delta y = \beta_1 \cdot \Delta x_1$$

- **In the Log Model:**
  The effect of x_1 on y is **proportional** to the current level of y. A **one-unit increase** in x_1 leads to a **percentage change** in y, approximately equal to $100 \times \beta_1\%$. Specifically:

$$\Delta \ln(y) = \beta_1 \cdot \Delta x_1 \quad \Rightarrow \quad \%\Delta y \approx 100 \times \beta_1$$

  - For small values of y, the absolute change is small.
  - For large values of y, the absolute change is larger, reflecting the multiplicative nature of the model.

Why We Cannot Compare R^2 or Adjusted R^2 Directly

- The level model explains variance in the original scale of y, while the log model explains variance in the logarithmic scale of y.
- The SST (Total Sum of Squares) differs across the models because the dependent variable is transformed, making direct comparisons of R^2 invalid.
- Adjusted R^2 does not resolve this issue because it also depends on the scale of the dependent variable.

Approach to Compare Model Fit Across Transformations

To compare models on the **same scale** as the original dependent variable (y), we need to **"un-transform"** the predictions from the log model. Here's the step-by-step procedure:

Step-by-Step Procedure

1. **Estimate the Log Model**
   Fit the log-linear model and obtain the predicted values:

$$\widehat{\ln(y)} = \hat{\beta}_0 + \hat{\beta}_1 x_1$$

2. **Un-Transform the Predictions**
   Convert the predicted values back to the original scale of y using the exponential function:

$$\hat{m} = \exp(\widehat{\ln(y)})$$

   - This transformation assumes that the errors are homoskedastic in the log model.
   - **Note:** To correct for potential bias due to Jensen's inequality, a **smearing estimator** can be applied, but for simplicity, we use the basic exponential transformation here.

3. **Fit a Regression Without an Intercept**
   Regress the actual y on the un-transformed predictions $\widehat{m}$ **without an intercept**:

$$y = \alpha \widehat{m} + u$$

   - The coefficient α adjusts for any scaling differences between the predicted and actual values.
   - The residual term u captures the unexplained variance.

4. **Compute the Scaled R^2**
   Calculate the **squared correlation** between the observed y and the predicted values $\hat{y}$ from the above regression:

$$R^2_{\text{scaled}} = \left(\text{Corr}(y, \hat{y})\right)^2$$

   - This **scaled** R^2 represents how well the log-transformed model predicts the original y on its natural scale.
   - Now, you can **compare** R^2_{scaled} from the log model with the **regular** R^2 from the level model.

Key Insights

- **If** R^2_{scaled} (from the log model) $>$ R^2 (from the level model): The log model fits the data better.
- **If** $R^2_{\text{scaled}} < R^2$ (from the level model): The level model provides a better fit.
- **If both are similar:** Consider other model diagnostics, theoretical justification, or model simplicity.

Caveats and Considerations

- **Heteroskedasticity:** If heteroskedasticity is present, the un-transformation may introduce bias.
- **Error Distribution:** Log-transformed models assume multiplicative errors, which may not be appropriate in all contexts.
- **Smearing Estimator (Advanced Correction):** To adjust for bias in the back-transformation, apply the smearing estimator:

$$\hat{y} = \exp(\widehat{\ln(y)}) \cdot \hat{S}$$

  Where $\hat{S}$ is the **mean of the exponentiated residuals** from the log model.

```r
# Install and load necessary libraries
# install.packages("nonnest2")  # Uncomment if not already installed
library(nonnest2)    # For Vuong Test
library(lmtest)      # For J-Test

# Simulated dataset
set.seed(123)
n <- 100
x <- rnorm(n, mean = 50, sd = 10)
z <- rnorm(n, mean = 100, sd = 20)
epsilon <- rnorm(n)

# Competing models (non-nested)
# Model A: Linear relationship with x
y <- 5 + 0.3 * x + epsilon
model_A <- lm(y ~ x)

# Model B: Log-linear relationship with z
model_B <- lm(y ~ log(z))

# ----------------------------------------------------------------------
# Vuong Test (Correct Function)
# ----------------------------------------------------------------------
vuong_test <- vuongtest(model_A, model_B)
print(vuong_test)
#>
#> Model 1
#>  Class: lm
#>  Call: lm(formula = y ~ x)
#>
#> Model 2
#>  Class: lm
#>  Call: lm(formula = y ~ log(z))
#>
#> Variance test
#>   H0: Model 1 and Model 2 are indistinguishable
#>   H1: Model 1 and Model 2 are distinguishable
#>     w2 = 0.681,   p = 2.35e-08
#>
#> Non-nested likelihood ratio test
#>   H0: Model fits are equal for the focal population
#>   H1A: Model 1 fits better than Model 2
#>     z = 13.108,   p = <2e-16
#>   H1B: Model 2 fits better than Model 1
#>     z = 13.108,   p = 1
```

```r
# ----------------------------------------------------------------------
# Davidson-MacKinnon J-Test
# ----------------------------------------------------------------------

# Step 1: Testing Model A against Model B
# Obtain fitted values from Model B
fitted_B <- fitted(model_B)

# Auxiliary regression: Add fitted_B to Model A
j_test_A_vs_B <- lm(y ~ x + fitted_B)
summary(j_test_A_vs_B)
#>
#> Call:
#> lm(formula = y ~ x + fitted_B)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -1.8717 -0.6573 -0.1223  0.6154  2.0952
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 14.70881   25.98307   0.566    0.573
#> x            0.28671    0.01048  27.358   <2e-16 ***
#> fitted_B    -0.43702    1.27500  -0.343    0.733
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.951 on 97 degrees of freedom
#> Multiple R-squared:  0.8854, Adjusted R-squared:  0.883
#> F-statistic: 374.5 on 2 and 97 DF,  p-value: < 2.2e-16

# Step 2: Testing Model B against Model A
# Obtain fitted values from Model A
fitted_A <- fitted(model_A)

# Auxiliary regression: Add fitted_A to Model B
j_test_B_vs_A <- lm(y ~ log(z) + fitted_A)
summary(j_test_B_vs_A)
#>
#> Call:
#> lm(formula = y ~ log(z) + fitted_A)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -1.8717 -0.6573 -0.1223  0.6154  2.0952
```

```
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.77868    2.39275  -0.325    0.746
#> log(z)       0.16829    0.49097   0.343    0.733
#> fitted_A     1.00052    0.03657  27.358   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.951 on 97 degrees of freedom
#> Multiple R-squared:  0.8854, Adjusted R-squared:  0.883
#> F-statistic: 374.5 on 2 and 97 DF,  p-value: < 2.2e-16
```

14.3 Heteroskedasticity Tests

Heteroskedasticity occurs when the variance of the error terms (ϵ_i) in a regression model is **not constant** across observations. This violates the Classical OLS Assumption, specifically the assumption of **homoskedasticity** (Assumption A4 Homoskedasticity in the Gauss-Markov Theorem), which states:

$$\text{Var}(\epsilon_i) = \sigma^2 \quad \forall\, i$$

When heteroskedasticity is present:

- Ordinary Least Squares estimators remain **unbiased** but become **inefficient** (i.e., no longer Best Linear Unbiased Estimators—BLUE).
- The **standard errors** of the estimates are biased, leading to unreliable **hypothesis tests** (e.g., t-tests and F-tests).

Detecting heteroskedasticity is crucial for ensuring the validity of regression results. This section covers key tests used to identify heteroskedasticity:

- Breusch–Pagan Test
- White Test
- Goldfeld–Quandt Test
- Park Test
- Glejser Test

14.3.1 Breusch–Pagan Test

The **Breusch–Pagan (BP) Test** is one of the most widely used tests for detecting heteroskedasticity (Breusch and Pagan, 1979). It examines whether

the variance of the residuals depends on the independent variables.

Hypotheses

- Null Hypothesis (H_0): Homoskedasticity ($\text{Var}(\epsilon_i) = \sigma^2$ is constant).
- Alternative Hypothesis (H_1): Heteroskedasticity exists; the variance of ϵ_i depends on the independent variables.

Procedure

1. **Estimate the original regression model:**

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_k x_{ki} + \epsilon_i$$

Obtain the residuals $\hat{\epsilon}_i$ from this regression.

2. **Compute the squared residuals:**

$$\hat{\epsilon}_i^2$$

3. **Auxiliary Regression:** Regress the squared residuals on the independent variables:

$$\hat{\epsilon}_i^2 = \alpha_0 + \alpha_1 x_{1i} + \alpha_2 x_{2i} + \cdots + \alpha_k x_{ki} + u_i$$

4. **Calculate the Test Statistic:**

The BP test statistic is:

$$\text{BP} = n \cdot R^2_{\text{aux}}$$

Where:

- n is the sample size,
- R^2_{aux} is the R^2 from the auxiliary regression.

5. **Decision Rule:**

- Under H_0, the BP statistic follows a **chi-squared distribution** with k degrees of freedom (where k is the number of independent variables):

$$\text{BP} \sim \chi^2_k$$

- **Reject** H_0 if the BP statistic exceeds the critical value from the chi-squared distribution.

Advantages and Limitations

- **Advantage:** Simple to implement; directly tests the relationship between residual variance and regressors.
- **Limitation:** Sensitive to non-normality; less effective when heteroskedasticity is not linearly related to independent variables.

14.3.2 White Test

The **White Test** is a more general heteroskedasticity test that does not require specifying the form of heteroskedasticity (White, 1980). It can detect both **linear** and **nonlinear** forms.

Hypotheses

- Null Hypothesis (H_0): Homoskedasticity.
- Alternative Hypothesis (H_1): Heteroskedasticity (of any form).

Procedure

1. **Estimate the original regression model** and obtain residuals $\hat{\epsilon}_i$.
2. **Auxiliary Regression:** Regress the squared residuals on:
   - The original independent variables $(x_{1i}, x_{2i}, \dots, x_{ki})$,
   - Their **squares** $(x_{1i}^2, x_{2i}^2, \dots, x_{ki}^2)$,
   - Their **cross-products** (e.g., $x_{1i}x_{2i}$).

   The auxiliary regression is:

$$\hat{\epsilon}_i^2 = \alpha_0 + \alpha_1 x_{1i} + \alpha_2 x_{2i} + \cdots + \alpha_k x_{ki} + \alpha_{k+1} x_{1i}^2 + \cdots + \alpha_{2k} x_{ki}^2 + \alpha_{2k+1}(x_{1i}x_{2i}) + u_i$$

3. **Calculate the Test Statistic:**

$$\text{White} = n \cdot R_{\text{aux}}^2$$

4. **Decision Rule:**

- Under H_0, the statistic follows a **chi-squared distribution** with degrees of freedom equal to the number of auxiliary regressors:

$$\text{White} \sim \chi_{\text{df}}^2$$

- **Reject** H_0 if the statistic exceeds the critical chi-squared value.

Advantages and Limitations

- **Advantage:** Can detect a wide range of heteroskedasticity patterns.
- **Limitation:** May suffer from overfitting in small samples due to many auxiliary regressors.

14.3.3 Goldfeld–Quandt Test

The **Goldfeld–Quandt Test** is a simple test that detects heteroskedasticity by comparing the variance of residuals in two different subsets of the data (Goldfeld and Quandt, 1965).

Hypotheses

- Null Hypothesis (H_0): Homoskedasticity.
- Alternative Hypothesis (H_1): Heteroskedasticity; variances differ between groups.

Procedure

1. **Sort the data** based on an independent variable suspected to cause heteroskedasticity.
2. **Split the data** into three groups:
   - **Group 1:** Lower values,
   - **Group 2:** Middle values (often omitted),
   - **Group 3:** Higher values.
3. **Estimate the regression model** separately for Groups 1 and 3. Obtain the residual sum of squares (SSR_1 and SSR_2).
4. **Calculate the Test Statistic:**

$$F = \frac{SSR_2/(n_2 - k)}{SSR_1/(n_1 - k)}$$

   Where:

   - n_1 and n_2 are the number of observations in Groups 1 and 3, respectively,
   - k is the number of estimated parameters.
5. **Decision Rule:**

- Under H_0, the test statistic follows an F-distribution with $(n_2 - k, n_1 - k)$ degrees of freedom:

$$F \sim F_{(n_2-k, n_1-k)}$$

- **Reject** H_0 if F exceeds the critical value.

Advantages and Limitations

- **Advantage:** Simple to apply when heteroskedasticity is suspected to vary systematically with an independent variable.
- **Limitation:** Requires arbitrary splitting of data and assumes the error variance changes abruptly between groups.

14.3.4 Park Test

The **Park Test** identifies heteroskedasticity by modeling the error variance as a function of an independent variable (Park, 1966).

Hypotheses

- Null Hypothesis (H_0): Homoskedasticity.
- Alternative Hypothesis (H_1): Heteroskedasticity; variance depends on an independent variable.

Procedure

1. **Estimate the original regression** and obtain residuals $\hat{\epsilon}_i$.
2. **Transform the residuals:** Take the natural logarithm of the squared residuals:

$$\ln(\hat{\epsilon}_i^2)$$

3. **Auxiliary Regression:** Regress $\ln(\hat{\epsilon}_i^2)$ on the independent variable(s):

$$\ln(\hat{\epsilon}_i^2) = \alpha_0 + \alpha_1 \ln(x_i) + u_i$$

4. **Decision Rule:**

- Test whether $\alpha_1 = 0$ using a t-test.
- **Reject** H_0 if α_1 is statistically significant, indicating heteroskedasticity.

Advantages and Limitations

- **Advantage:** Simple to implement; works well when the variance follows a log-linear relationship.
- **Limitation:** Assumes a specific functional form for the variance, which may not hold in practice.

14.3.5 Glejser Test

The **Glejser Test** detects heteroskedasticity by regressing the **absolute value of residuals** on the independent variables (Glejser, 1969).

Hypotheses

- Null Hypothesis (H_0): Homoskedasticity.
- Alternative Hypothesis (H_1): Heteroskedasticity exists.

Procedure

1. **Estimate the original regression** and obtain residuals $\hat{\epsilon}_i$.
2. **Auxiliary Regression:** Regress the absolute residuals on the independent variables:

$$|\hat{\epsilon}_i| = \alpha_0 + \alpha_1 x_{1i} + \alpha_2 x_{2i} + \cdots + \alpha_k x_{ki} + u_i$$

3. **Decision Rule:**

- Test the significance of the coefficients $(\alpha_1, \alpha_2, ...)$ using t-tests.
- **Reject** H_0 if any coefficient is statistically significant, indicating heteroskedasticity.

Advantages and Limitations

- **Advantage:** Flexible; can detect various forms of heteroskedasticity.
- **Limitation:** Sensitive to outliers since it relies on absolute residuals.

14.3.6 Summary of Heteroskedasticity Tests

Test	Type	Assumptions	Key Statistic	When to Use
Breusch–Pagan	Parametric	Linear relationship with predictors	χ^2	General-purpose test
White	General (non-parametric)	No functional form assumption	χ^2	Detects both linear & nonlinear forms
Goldfeld–Quandt	Group comparison	Assumes known ordering of variance	F-distribution	When heteroskedasticity varies by groups
Park	Parametric (log-linear)	Assumes log-linear variance	t-test	When variance depends on predictors
Glejser	Parametric	Based on absolute residuals	t-test	Simple test for variance dependence

Detecting heteroskedasticity is critical for ensuring the reliability of regression models. While each test has strengths and limitations, combining multiple tests can provide robust insights. Once heteroskedasticity is detected, consider using **robust standard errors** or alternative estimation techniques (e.g., Generalized Least Squares or Weighted Least Squares) to address the issue.

```r
# Install and load necessary libraries
# install.packages("lmtest")      # For Breusch-Pagan Test
# install.packages("car")         # For additional regression diagnostics
# install.packages("sandwich")    # For robust covariance estimation


library(lmtest)
library(car)
library(sandwich)

# Simulated dataset
set.seed(123)
n <- 100
x1 <- rnorm(n, mean = 50, sd = 10)
x2 <- rnorm(n, mean = 30, sd = 5)
epsilon <-
    rnorm(n, sd = x1 * 0.1)  # Heteroskedastic errors increasing with x1
```

```r
y <- 5 + 0.4 * x1 - 0.3 * x2 + epsilon

# Original regression model
model <- lm(y ~ x1 + x2)

# ----------------------------------------------------------------------
# 1. Breusch-Pagan Test
# ----------------------------------------------------------------------
# Null Hypothesis: Homoskedasticity
bp_test <- bptest(model)
print(bp_test)
#>
#>  studentized Breusch-Pagan test
#>
#> data:  model
#> BP = 7.8141, df = 2, p-value = 0.0201

# ----------------------------------------------------------------------
# 2. White Test (using Breusch-Pagan framework with squares & interactions)
# ----------------------------------------------------------------------
# Create squared and interaction terms
model_white <-
    lm(residuals(model) ^ 2 ~ x1 + x2 + I(x1 ^ 2) + I(x2 ^ 2) + I(x1 * x2))
white_statistic <-
    summary(model_white)$r.squared * n  # White Test Statistic
df_white <-
    length(coef(model_white)) - 1              # Degrees of freedom
p_value_white <- 1 - pchisq(white_statistic, df_white)

# Display White Test result
cat("White Test Statistic:", white_statistic, "\n")
#> White Test Statistic: 11.85132
cat("Degrees of Freedom:", df_white, "\n")
#> Degrees of Freedom: 5
cat("P-value:", p_value_white, "\n")
#> P-value: 0.0368828

# ----------------------------------------------------------------------
# 3. Goldfeld-Quandt Test
# ----------------------------------------------------------------------
# Null Hypothesis: Homoskedasticity
# Sort data by x1 (suspected source of heteroskedasticity)
gq_test <-
    gqtest(model, order.by = ~ x1, fraction = 0.2)  # Omit middle 20% of data
print(gq_test)
```

```
#>
#>  Goldfeld-Quandt test
#>
#> data:  model
#> GQ = 1.8352, df1 = 37, df2 = 37, p-value = 0.03434
#> alternative hypothesis: variance increases from segment 1 to 2

# ----------------------------------------------------------------------
# 4. Park Test
# ----------------------------------------------------------------------
# Step 1: Get residuals and square them
residuals_squared <- residuals(model) ^ 2

# Step 2: Log-transform squared residuals
log_residuals_squared <- log(residuals_squared)

# Step 3: Regress log(residuals^2) on log(x1) (assuming variance depends on x1)
park_test <- lm(log_residuals_squared ~ log(x1))
summary(park_test)
#>
#> Call:
#> lm(formula = log_residuals_squared ~ log(x1))
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -9.3633 -1.3424  0.4218  1.6089  3.0697
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  -1.6319     4.5982  -0.355    0.723
#> log(x1)       0.8903     1.1737   0.759    0.450
#>
#> Residual standard error: 2.171 on 98 degrees of freedom
#> Multiple R-squared:  0.005837,   Adjusted R-squared:  -0.004308
#> F-statistic: 0.5754 on 1 and 98 DF,  p-value: 0.4499

# ----------------------------------------------------------------------
# 5. Glejser Test
# ----------------------------------------------------------------------
# Step 1: Absolute value of residuals
abs_residuals <- abs(residuals(model))

# Step 2: Regress absolute residuals on independent variables
glejser_test <- lm(abs_residuals ~ x1 + x2)
summary(glejser_test)
```

```
#>
#> Call:
#> lm(formula = abs_residuals ~ x1 + x2)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -4.3096 -2.2680 -0.4564  1.9554  8.3921
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  0.755846   2.554842   0.296   0.7680
#> x1           0.064896   0.032852   1.975   0.0511 .
#> x2          -0.008495   0.062023  -0.137   0.8913
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 2.98 on 97 degrees of freedom
#> Multiple R-squared:  0.0392, Adjusted R-squared:  0.01939
#> F-statistic: 1.979 on 2 and 97 DF,  p-value: 0.1438
```

Interpretation of the Results

1. **Breusch–Pagan Test**
   - **Null Hypothesis** (H_0): Homoskedasticity (constant error variance).
   - **Alternative Hypothesis** (H_1): Heteroskedasticity exists (error variance depends on predictors).
   - **Decision Rule:**
     - **Reject** H_0 if *p-value* $< 0.05 \rightarrow$ Evidence of heteroskedasticity.
     - **Fail to reject** H_0 if *p-value* $\geq 0.05 \rightarrow$ No strong evidence of heteroskedasticity.
2. **White Test**
   - **Null Hypothesis** (H_0): Homoskedasticity.
   - **Alternative Hypothesis** (H_1): Heteroskedasticity (of any form, linear or nonlinear).
   - **Decision Rule:**
     - **Reject** H_0 if *p-value* $< 0.05 \rightarrow$ Presence of heteroskedasticity.
     - **Fail to reject** H_0 if *p-value* $\geq 0.05 \rightarrow$ Homoskedasticity likely holds.
3. **Goldfeld–Quandt Test**

- **Null Hypothesis (H_0)**: Homoskedasticity (equal variances across groups).
- **Alternative Hypothesis (H_1)**: Heteroskedasticity (unequal variances between groups).
- **Decision Rule:**
  - **Reject** H_0 if *p-value* $< 0.05 \rightarrow$ Variances differ between groups, indicating heteroskedasticity.
  - **Fail to reject** H_0 if *p-value* $\geq 0.05 \rightarrow$ No significant evidence of heteroskedasticity.

4. **Park Test**
   - **Null Hypothesis (H_0)**: No relationship between the variance of errors and predictor(s) (homoskedasticity).
   - **Alternative Hypothesis (H_1)**: Variance of errors depends on predictor(s).
   - **Decision Rule:**
     - **Reject** H_0 if the *coefficient of* $\log(x_1)$ *is statistically significant* (*p-value* < 0.05).
     - **Fail to reject** H_0 if *p-value* ≥ 0.05.
5. **Glejser Test**
   - **Null Hypothesis (H_0)**: Homoskedasticity (no relationship between absolute residuals and predictors).
   - **Alternative Hypothesis (H_1)**: Heteroskedasticity exists.
   - **Decision Rule:**
     - **Reject** H_0 if any predictor is statistically significant (*p-value* < 0.05).
     - **Fail to reject** H_0 if *p-value* ≥ 0.05.

14.4 Functional Form Tests

Functional form misspecification occurs when the chosen regression model does not correctly represent the true relationship between the dependent and independent variables. This can happen due to:

- **Omitted variables** (important predictors not included),
- **Incorrect transformations** of variables (e.g., missing nonlinear relationships),

- **Incorrect interaction terms** (missing interaction effects between variables),
- **Inappropriate linearity assumptions**.

Functional form errors can lead to **biased and inconsistent** estimators, undermining the validity of statistical inferences. To detect such issues, several diagnostic tests are available.

Key Functional Form Tests:

1. Ramsey RESET Test (Regression Equation Specification Error Test)
2. Harvey–Collier Test
3. Rainbow Test

Each test focuses on identifying different aspects of potential model misspecification.

14.4.1 Ramsey RESET Test (Regression Equation Specification Error Test)

The **Ramsey RESET Test** is one of the most widely used tests to detect functional form misspecification (Ramsey, 1969). It examines whether adding **nonlinear transformations** of the fitted values (or regressors) improves the model fit.

Hypotheses

- Null Hypothesis (H_0): The model is correctly specified.
- Alternative Hypothesis (H_1): The model suffers from omitted variables, incorrect functional form, or other specification errors.

Procedure

1. **Estimate the original regression model:**

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_k x_{ki} + \epsilon_i$$

2. **Obtain the fitted values:**

$$\hat{y}_i$$

3. **Augment the model with powers of the fitted values (squared, cubed, etc.):**

$$y_i = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_k x_{ki} + \gamma_1 \hat{y}_i^2 + \gamma_2 \hat{y}_i^3 + u_i$$

4. **Test the joint significance** of the added terms:

$$H_0 : \gamma_1 = \gamma_2 = 0$$

5. **Compute the F-statistic:**

$$F = \frac{(SSR_{\text{restricted}} - SSR_{\text{unrestricted}})/q}{SSR_{\text{unrestricted}}/(n-k-q-1)}$$

Where:

- $SSR_{\text{restricted}}$ = Sum of Squared Residuals from the original model,
- $SSR_{\text{unrestricted}}$ = SSR from the augmented model,
- q = Number of additional terms (e.g., 2 if adding $\hat{y}^2$ and $\hat{y}^3$),
- n = Sample size,
- k = Number of predictors in the original model.

Decision Rule

- Under H_0, the F-statistic follows an F-distribution with $(q, n-k-q-1)$ degrees of freedom.
- Reject H_0 if the F-statistic exceeds the critical value, indicating functional form misspecification.

Advantages and Limitations

- **Advantage:** Simple to implement; detects omitted variables and incorrect functional forms.
- **Limitation:** Does not identify which variable or functional form is incorrect—only indicates the presence of an issue.

14.4.2 Harvey–Collier Test

The **Harvey–Collier Test** evaluates whether the model's residuals display systematic patterns, which would indicate functional form misspecification (Harvey and Collier, 1977). It is based on testing for a **non-zero mean** in the residuals after projection onto specific components.

Hypotheses

- Null Hypothesis (H_0): The model is correctly specified (residuals are random noise with zero mean).
- Alternative Hypothesis (H_1): The model is misspecified (residuals contain systematic patterns).

Procedure

1. **Estimate the original regression model** and obtain residuals $\hat{\epsilon}_i$.
2. **Project the residuals** onto the space spanned by a specially constructed test vector (often derived from the inverse of the design matrix in linear regression).
3. **Calculate the Harvey–Collier statistic:**

$$t = \frac{\bar{\epsilon}}{\mathrm{SE}(\bar{\epsilon})}$$

   Where:

   - $\bar{\epsilon}$ is the mean of the projected residuals,
   - $\mathrm{SE}(\bar{\epsilon})$ is the standard error of the mean residual.

Decision Rule:

- The test statistic follows a t-distribution under H_0.
- **Reject** H_0 if the t-statistic is significantly different from zero.

Advantages and Limitations

- **Advantage:** Simple to apply and interpret; good for detecting subtle misspecifications.
- **Limitation:** Sensitive to outliers; may have reduced power in small samples.

14.4.3 Rainbow Test

The **Rainbow Test** is a general-purpose diagnostic tool for functional form misspecification (Utts, 1982). It compares the performance of the model on the **full sample** versus a **central subsample**, where the central subsample contains observations near the median of the independent variables.

Hypotheses

- Null Hypothesis (H_0): The model is correctly specified.
- Alternative Hypothesis (H_1): The model is misspecified.

Procedure

1. **Estimate the regression model** on the full dataset and record the residuals.

2. **Identify a central subsample** (e.g., observations near the median of key predictors).
3. **Estimate the model again** on the central subsample.
4. **Compare the predictive accuracy** between the full sample and subsample using an F-statistic:

$$F = \frac{(SSR_{\text{full}} - SSR_{\text{subsample}})/q}{SSR_{\text{subsample}}/(n-k-q)}$$

Where q is the number of restrictions implied by using the subsample.

Decision Rule

- Under H_0, the test statistic follows an F-distribution.
- Reject H_0 if the F-statistic is significant, indicating model misspecification.

Advantages and Limitations

- **Advantage:** Robust to various forms of misspecification.
- **Limitation:** Choice of subsample may influence results; less informative about the specific nature of the misspecification.

14.4.4 Summary of Functional Form Tests

Test	Type	Key Statistic	Purpose	When to Use
Ramsey RESET Test	Augmented regression	F-test	Detects omitted variables, nonlinearities	General model specification testing
Harvey–Collier Test	Residual-based	t-test	Detects systematic patterns in residuals	Subtle misspecifications in linear models
Rainbow Test	Subsample comparison	F-test	Tests model stability across subsamples	Comparing central vs. full sample

Functional form misspecification can severely distort regression results, leading to biased estimates and invalid inferences. While no single test can detect

all types of misspecification, using a combination of tests provides a robust framework for model diagnostics.

```
# Install and load necessary libraries
# install.packages("lmtest")      # For RESET and Harvey-Collier Test
# install.packages("car")         # For diagnostic tests
# install.packages("strucchange") # For Rainbow Test

library(lmtest)
library(car)
library(strucchange)

# Simulated dataset
set.seed(123)
n <- 100
x1 <- rnorm(n, mean = 50, sd = 10)
x2 <- rnorm(n, mean = 30, sd = 5)
epsilon <- rnorm(n)
y <- 5 + 0.4 * x1 - 0.3 * x2 + epsilon

# Original regression model
model <- lm(y ~ x1 + x2)

# ----------------------------------------------------------------------
# 1. Ramsey RESET Test
# ----------------------------------------------------------------------
# Null Hypothesis: The model is correctly specified
reset_test <-
    resettest(model, power = 2:3, type = "fitted")  # Adds ŷ² and ŷ³
print(reset_test)
#>
#>  RESET test
#>
#> data:  model
#> RESET = 0.1921, df1 = 2, df2 = 95, p-value = 0.8255

# ----------------------------------------------------------------------
# 2. Harvey-Collier Test
# ----------------------------------------------------------------------
# Null Hypothesis: The model is correctly specified (residuals have zero mean)
hc_test <- harvtest(model)
print(hc_test)
#>
#>  Harvey-Collier test
#>
#> data:  model
```

```
#> HC = 0.041264, df = 96, p-value = 0.9672

# ----------------------------------------------------------------------
# 3. Rainbow Test
# ----------------------------------------------------------------------
# Null Hypothesis: The model is correctly specified
rainbow_test <- lmtest::raintest (model)
print(rainbow_test)
#>
#>  Rainbow test
#>
#> data:  model
#> Rain = 1.1857, df1 = 50, df2 = 47, p-value = 0.279
```

Interpretation of the Results

1. **Ramsey RESET Test (Regression Equation Specification Error Test)**
   - **Null Hypothesis** (H_0): The model is correctly specified.
   - **Alternative Hypothesis** (H_1): The model suffers from omitted variables, incorrect functional form, or other specification errors.
   - **Decision Rule:**
     - **Reject** H_0 if *p-value* $< 0.05 \rightarrow$ Evidence of model misspecification (e.g., missing nonlinear terms).
     - **Fail to reject** H_0 if *p-value* $\geq 0.05 \rightarrow$ No strong evidence of misspecification.
2. **Harvey–Collier Test**
   - **Null Hypothesis** (H_0): The model is correctly specified (residuals are random noise with zero mean).
   - **Alternative Hypothesis** (H_1): The model is misspecified (residuals contain systematic patterns).
   - **Decision Rule:**
     - **Reject** H_0 if *p-value* $< 0.05 \rightarrow$ Model misspecification detected (non-random residual patterns).
     - **Fail to reject** H_0 if *p-value* $\geq 0.05 \rightarrow$ No evidence of misspecification.
3. **Rainbow Test**
   - **Null Hypothesis** (H_0): The model is correctly specified.
   - **Alternative Hypothesis** (H_1): The model is misspecified.

- **Decision Rule:**
  - **Reject** H_0 if *p-value* $< 0.05 \rightarrow$ Evidence of model misspecification (model performs differently on subsets).
  - **Fail to reject** H_0 if *p-value* $\geq 0.05 \rightarrow$ Model specification appears valid.

14.5 Autocorrelation Tests

Autocorrelation (also known as **serial correlation**) occurs when the error terms (ϵ_t) in a regression model are **correlated across observations**, violating the assumption of **independence** in the classical linear regression model. This issue is particularly common in **time-series data**, where observations are ordered over time.

Consequences of Autocorrelation:

- OLS estimators remain unbiased but become **inefficient**, meaning they do not have the minimum variance among all linear unbiased estimators.
- **Standard errors** are biased, leading to unreliable hypothesis tests (e.g., t-tests and F-tests).
- Potential **underestimation of standard errors**, increasing the risk of Type I errors (false positives).

14.5.1 Durbin–Watson Test

The **Durbin–Watson (DW) Test** is the most widely used test for detecting **first-order autocorrelation**, where the current error term is correlated with the previous one:

$$\epsilon_t = \rho\, \epsilon_{t-1} + u_t$$

Where:

- ρ is the autocorrelation coefficient,
- u_t is a white noise error term.

Hypotheses

- Null Hypothesis (H_0): No first-order autocorrelation ($\rho = 0$).
- Alternative Hypothesis (H_1): First-order autocorrelation exists ($\rho \neq 0$).

Durbin–Watson Test Statistic

The DW statistic is calculated as:

$$DW = \frac{\sum_{t=2}^{n}(\hat{\epsilon}_t - \hat{\epsilon}_{t-1})^2}{\sum_{t=1}^{n}\hat{\epsilon}_t^2}$$

Where:

- $\hat{\epsilon}_t$ are the residuals from the regression,
- n is the number of observations.

Decision Rule

- The DW statistic ranges from **0 to 4**:
  - **DW 2**: No autocorrelation.
  - **DW < 2**: Positive autocorrelation.
  - **DW > 2**: Negative autocorrelation.

For more precise interpretation:

- Use **Durbin–Watson critical values** (d_L and d_U) to form decision boundaries.
- If the test statistic falls into an **inconclusive range**, consider alternative tests like the Breusch–Godfrey test.

Advantages and Limitations

- **Advantage:** Simple to compute; specifically designed for detecting first-order autocorrelation.
- **Limitation:** Inconclusive in some cases; **invalid** when lagged dependent variables are included in the model.

14.5.2 Breusch–Godfrey Test

The **Breusch–Godfrey (BG) Test** is a more general approach that can detect **higher-order autocorrelation** (e.g., second-order, third-order) and is valid even when **lagged dependent variables** are included in the model (Breusch, 1978; Godfrey, 1978).

Hypotheses

- Null Hypothesis (H_0): No autocorrelation of any order (up to lag p).

- Alternative Hypothesis (H_1): Autocorrelation exists at some lag(s).

Procedure

1. **Estimate the original regression model** and obtain residuals $\hat{\epsilon}_t$:

$$y_t = \beta_0 + \beta_1 x_{1t} + \cdots + \beta_k x_{kt} + \epsilon_t$$

2. **Run an auxiliary regression** by regressing the residuals on the original regressors plus p lagged residuals:

$$\hat{\epsilon}_t = \alpha_0 + \alpha_1 x_{1t} + \cdots + \alpha_k x_{kt} + \rho_1 \hat{\epsilon}_{t-1} + \cdots + \rho_p \hat{\epsilon}_{t-p} + u_t$$

3. **Calculate the test statistic:**

$$\text{BG} = n \cdot R^2_{\text{aux}}$$

   Where:

   - n is the sample size,
   - R^2_{aux} is the R^2 from the auxiliary regression.

Decision Rule:

- Under H_0, the BG statistic follows a **chi-squared distribution** with p degrees of freedom:

$$\text{BG} \sim \chi^2_p$$

- **Reject** H_0 if the statistic exceeds the critical chi-squared value, indicating the presence of autocorrelation.

Advantages and Limitations

- **Advantage:** Can detect **higher-order autocorrelation**; valid with lagged dependent variables.
- **Limitation:** More computationally intensive than the Durbin–Watson test.

14.5.3 Ljung–Box Test (or Box–Pierce Test)

The **Ljung–Box Test** is a **portmanteau test** designed to detect **autocorrelation at multiple lags** simultaneously (Box and Pierce, 1970; Ljung and Box, 1978). It is commonly used in **time-series analysis** to check residual autocorrelation after model estimation (e.g., in ARIMA models).

Hypotheses

- Null Hypothesis (H_0): No autocorrelation up to lag h.
- Alternative Hypothesis (H_1): Autocorrelation exists at one or more lags.

Ljung–Box Test Statistic

The Ljung–Box statistic is calculated as:

$$Q = n(n+2)\sum_{k=1}^{h}\frac{\hat{\rho}_k^2}{n-k}$$

Where:

- n = Sample size,
- h = Number of lags tested,
- $\hat{\rho}_k$ = Sample autocorrelation at lag k.

Decision Rule

- Under H_0, the Q statistic follows a **chi-squared distribution** with h degrees of freedom:

$$Q \sim \chi_h^2$$

- **Reject** H_0 if Q exceeds the critical value, indicating significant autocorrelation.

Advantages and Limitations

- **Advantage:** Detects autocorrelation across multiple lags simultaneously.
- **Limitation:** Less powerful for detecting specific lag structures; sensitive to model misspecification.

14.5.4 Runs Test

The **Runs Test** is a **non-parametric test** that examines the randomness of residuals. It is based on the number of **runs**—sequences of consecutive residuals with the same sign.

Hypotheses

- Null Hypothesis (H_0): Residuals are randomly distributed (no autocorrelation).
- Alternative Hypothesis (H_1): Residuals exhibit non-random patterns (indicating autocorrelation).

Procedure

1. **Classify residuals** as positive or negative.
2. **Count the number of runs:** A **run** is a sequence of consecutive positive or negative residuals.
3. **Calculate the expected number of runs under randomness:**

$$E(R) = \frac{2n_+ n_-}{n} + 1$$

   Where:

   - n_+ = Number of positive residuals,
   - n_- = Number of negative residuals,
   - $n = n_+ + n_-$.

4. **Compute the test statistic (Z-score):**

$$Z = \frac{R - E(R)}{\sqrt{\mathrm{Var}(R)}}$$

   Where $\mathrm{Var}(R)$ is the variance of the number of runs under the null hypothesis.

Decision Rule

- Under H_0, the Z-statistic follows a **standard normal distribution**:

$$Z \sim N(0, 1)$$

- **Reject** H_0 if $|Z|$ exceeds the critical value from the standard normal distribution.

Advantages and Limitations

- **Advantage:** Non-parametric; does not assume normality or linearity.
- **Limitation:** Less powerful than parametric tests; primarily useful as a supplementary diagnostic.

14.5.5 Summary of Autocorrelation Tests

Test	Type	Key Statistic	Detects	When to Use
Durbin–Watson	Parametric	DW	First-order autocorrelation	Simple linear models without lagged dependent variables
Breusch–Godfrey	Parametric (general)	χ^2	Higher-order autocorrelation	Models with lagged dependent variables
Ljung–Box	Portmanteau (global test)	χ^2	Autocorrelation at multiple lags	Time-series models (e.g., ARIMA)
Runs Test	Non-parametric	Z-statistic	Non-random patterns in residuals	Supplementary diagnostic for randomness

Detecting autocorrelation is crucial for ensuring the efficiency and reliability of regression models, especially in time-series analysis. While the Durbin–Watson Test is suitable for detecting first-order autocorrelation, the Breusch–Godfrey Test and Ljung–Box Test offer more flexibility for higher-order and multi-lag dependencies. Non-parametric tests like the Runs Test serve as useful supplementary diagnostics.

```
# Install and load necessary libraries
# install.packages("lmtest")  # For Durbin-Watson and Breusch-Godfrey Tests
# install.packages("tseries") # For Runs Test
# install.packages("forecast")# For Ljung-Box Test
```

```r
library(lmtest)
library(tseries)
library(forecast)

# Simulated time-series dataset
set.seed(123)
n <- 100
time <- 1:n
x1 <- rnorm(n, mean = 50, sd = 10)
x2 <- rnorm(n, mean = 30, sd = 5)

# Introducing autocorrelation in errors
epsilon <- arima.sim(model = list(ar = 0.6), n = n)
# AR(1) process with   = 0.6
y <- 5 + 0.4 * x1 - 0.3 * x2 + epsilon

# Original regression model
model <- lm(y ~ x1 + x2)

# ----------------------------------------------------------------------
# 1. Durbin-Watson Test
# ----------------------------------------------------------------------
# Null Hypothesis: No first-order autocorrelation
dw_test <- dwtest(model)
print(dw_test)
#>
#>  Durbin-Watson test
#>
#> data:  model
#> DW = 0.77291, p-value = 3.323e-10
#> alternative hypothesis: true autocorrelation is greater than 0

# ----------------------------------------------------------------------
# 2. Breusch-Godfrey Test
# ----------------------------------------------------------------------
# Null Hypothesis: No autocorrelation up to lag 2
bg_test <- bgtest(model, order = 2)  # Testing for autocorrelation up to lag 2
print(bg_test)
#>
#>  Breusch-Godfrey test for serial correlation of order up to 2
#>
#> data:  model
#> LM test = 40.314, df = 2, p-value = 1.762e-09

# ----------------------------------------------------------------------
```

```r
# 3. Ljung-Box Test
# ----------------------------------------------------------------------
# Null Hypothesis: No autocorrelation up to lag 10
ljung_box_test <- Box.test(residuals(model), lag = 10, type = "Ljung-Box")
print(ljung_box_test)
#>
#>  Box-Ljung test
#>
#> data:  residuals(model)
#> X-squared = 50.123, df = 10, p-value = 2.534e-07

# ----------------------------------------------------------------------
# 4. Runs Test (Non-parametric)
# ----------------------------------------------------------------------
# Null Hypothesis: Residuals are randomly distributed
runs_test <- runs.test(as.factor(sign(residuals(model))))
print(runs_test)
#>
#>  Runs Test
#>
#> data:  as.factor(sign(residuals(model)))
#> Standard Normal = -4.2214, p-value = 2.428e-05
#> alternative hypothesis: two.sided
```

Interpretation of the Results

1. **Durbin–Watson Test**
   - **Null Hypothesis** (H_0): No first-order autocorrelation ($\rho = 0$).
   - **Alternative Hypothesis** (H_1): First-order autocorrelation exists ($\rho \neq 0$).
   - **Decision Rule:**
     - **Reject** H_0 if $DW < 1.5$ (positive autocorrelation) or $DW > 2.5$ (negative autocorrelation).
     - **Fail to reject** H_0 if $DW \approx 2$, suggesting no significant autocorrelation.
2. **Breusch–Godfrey Test**
   - **Null Hypothesis** (H_0): No autocorrelation up to lag p (here, $p = 2$).
   - **Alternative Hypothesis** (H_1): Autocorrelation exists at one or more lags.
   - **Decision Rule:**

  - **Reject** H_0 if *p-value* < 0.05, indicating significant autocorrelation.
  - **Fail to reject** H_0 if *p-value* ≥ 0.05, suggesting no evidence of autocorrelation.

3. **Ljung–Box Test**
   - **Null Hypothesis (H_0)**: No autocorrelation up to lag h (here, $h = 10$).
   - **Alternative Hypothesis (H_1)**: Autocorrelation exists at one or more lags.
   - **Decision Rule:**
     - **Reject** H_0 if *p-value* < 0.05, indicating significant autocorrelation.
     - **Fail to reject** H_0 if *p-value* ≥ 0.05, suggesting no evidence of autocorrelation.
4. **Runs Test (Non-parametric)**
   - **Null Hypothesis (H_0)**: Residuals are randomly distributed (no autocorrelation).
   - **Alternative Hypothesis (H_1)**: Residuals exhibit non-random patterns (indicating autocorrelation).
   - **Decision Rule:**
     - **Reject** H_0 if *p-value* < 0.05, indicating non-randomness and potential autocorrelation.
     - **Fail to reject** H_0 if *p-value* ≥ 0.05, suggesting randomness in residuals.

14.6 Multicollinearity Diagnostics

Multicollinearity occurs when two or more independent variables in a regression model are **highly correlated**, leading to several issues:

- **Unstable coefficient estimates:** Small changes in the data can cause large fluctuations in parameter estimates.
- **Inflated standard errors:** Reduces the precision of estimated coefficients, making it difficult to determine the significance of predictors.
- **Difficulty in assessing variable importance:** It becomes challenging to isolate the effect of individual predictors on the dependent variable.

Multicollinearity does not affect the overall fit of the model (e.g., R^2 remains high), but it distorts the reliability of individual coefficient estimates.

Key Multicollinearity Diagnostics:

1. Variance Inflation Factor
2. Tolerance Statistic
3. Condition Index and Eigenvalue Decomposition
4. Pairwise Correlation Matrix
5. Determinant of the Correlation Matrix

14.6.1 Variance Inflation Factor

The **Variance Inflation Factor (VIF)** is the most commonly used diagnostic for detecting multicollinearity. It measures how much the **variance of an estimated regression coefficient** is inflated due to multicollinearity compared to when the predictors are uncorrelated.

For each predictor X_j, the VIF is defined as:

$$\text{VIF}_j = \frac{1}{1 - R_j^2}$$

Where:

- R_j^2 is the **coefficient of determination** obtained by regressing X_j on all other independent variables in the model.

Interpretation of VIF

- **VIF = 1:** No multicollinearity (perfect independence).
- **1 < VIF < 5:** Moderate correlation, typically not problematic.
- **VIF 5:** High correlation; consider investigating further.
- **VIF 10:** Severe multicollinearity; corrective action is recommended.

Procedure

1. Regress each independent variable (X_j) on the remaining predictors.
2. Compute R_j^2 for each regression.
3. Calculate $\text{VIF}_j = 1/(1 - R_j^2)$.
4. Analyze VIF values to identify problematic predictors.

Advantages and Limitations

- **Advantage:** Easy to compute and interpret.

- **Limitation:** Detects only **linear** relationships; may not capture complex multicollinearity patterns involving multiple variables simultaneously.

14.6.2 Tolerance Statistic

The **Tolerance Statistic** is the **reciprocal of the VIF** and measures the proportion of variance in an independent variable **not explained** by the other predictors.

$$\text{Tolerance}_j = 1 - R_j^2$$

Where R_j^2 is defined as in the VIF calculation.

Interpretation of Tolerance

- **Tolerance close to 1:** Low multicollinearity.
- **Tolerance $<$ 0.2:** Potential multicollinearity problem.
- **Tolerance $<$ 0.1:** Severe multicollinearity.

Since **low tolerance** implies **high VIF**, both metrics provide consistent information.

Advantages and Limitations

- **Advantage:** Provides an intuitive measure of how much variance is "free" from multicollinearity.
- **Limitation:** Similar to VIF, focuses on linear dependencies.

14.6.3 Condition Index and Eigenvalue Decomposition

The **Condition Index** is a more advanced diagnostic that detects **multicollinearity involving multiple variables** simultaneously. It is based on the **eigenvalues** of the scaled independent variable matrix.

1. **Compute the scaled design matrix** $X'X$, where X is the matrix of independent variables.

2. **Perform eigenvalue decomposition** to obtain the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_k$.

3. **Calculate the Condition Index:**

$$\mathrm{CI}_j = \sqrt{\frac{\lambda_{\max}}{\lambda_j}}$$

Where:

- $\lambda_{\max}$ is the largest eigenvalue,
- λ_j is the j-th eigenvalue.

Interpretation of Condition Index

- **CI $<$ 10:** No serious multicollinearity.
- **10 CI $<$ 30:** Moderate to strong multicollinearity.
- **CI 30:** Severe multicollinearity.

A **high condition index** indicates near-linear dependence among variables.

Variance Decomposition Proportions

To identify which variables contribute to multicollinearity:

- Compute the **Variance Decomposition Proportions (VDP)** for each coefficient across eigenvalues.
- If **two or more variables** have high VDPs (e.g., > 0.5) associated with a **high condition index**, this indicates severe multicollinearity.

Advantages and Limitations

- **Advantage:** Detects multicollinearity involving **multiple variables**, which VIF may miss.
- **Limitation:** Requires matrix algebra knowledge; less intuitive than VIF or tolerance.

14.6.4 Pairwise Correlation Matrix

A **Pairwise Correlation Matrix** provides a simple diagnostic by computing the correlation coefficients between each pair of independent variables.

For variables X_i and X_j, the **correlation coefficient** is:

$$\rho_{ij} = \frac{\mathrm{Cov}(X_i, X_j)}{\sigma_{X_i}\sigma_{X_j}}$$

Where:

- $\text{Cov}(X_i, X_j)$ is the covariance,
- σ_{X_i} and σ_{X_j} are standard deviations.

Interpretation of Correlation Coefficients

- $|\rho| < 0.5$: Weak correlation (unlikely to cause multicollinearity).
- $0.5 \leq |\rho| < 0.8$: Moderate correlation; monitor carefully.
- $|\rho| \geq 0.8$: Strong correlation; potential multicollinearity issue.

Advantages and Limitations

- **Advantage:** Quick and easy to compute; useful for initial screening.
- **Limitation:** Detects **only pairwise relationships**; may miss multicollinearity involving more than two variables.

14.6.5 Determinant of the Correlation Matrix

The **Determinant of the Correlation Matrix** provides a global measure of multicollinearity. A **small determinant** indicates high multicollinearity.

1. Form the **correlation matrix** R of the independent variables.
2. Compute the **determinant**:

$$\det(R)$$

Interpretation

- $\det(R) \approx 1$: No multicollinearity (perfect independence).
- $\det(R) \approx 0$: Severe multicollinearity.

A determinant **close to zero** suggests that the correlation matrix is nearly singular, indicating strong multicollinearity.

Advantages and Limitations

- **Advantage:** Provides a single summary statistic for overall multicollinearity.
- **Limitation:** Does not indicate **which** variables are causing the problem.

14.6.6 Summary of Multicollinearity Diagnostics

Diagnostic	Type	Key Metric	Threshold for Concern	When to Use
Variance Inflation Factor	Parametric	$VIF = \frac{1}{1-R_j^2}$	$VIF \geq 5$ (moderate), $VIF \geq 10$ (severe)	General-purpose detection
Tolerance Statistic	Parametric	$1 - R_j^2$	< 0.2 (moderate), < 0.1 (severe)	Reciprocal of VIF for variance interpretation
Condition Index	Eigenvalue-based	$\sqrt{\frac{\lambda_{\max}}{\lambda_j}}$	> 10 (moderate), > 30 (severe)	Detects multicollinearity among multiple variables
Pairwise Correlation Matrix	Correlation-based	Pearson correlation (ρ)	$\lvert\rho\rvert \geq 0.8$	Initial screening for bivariate correlations
Determinant of Correlation Matrix	Global diagnostic	$\det(R)$	≈ 0 indicates severe multicollinearity	Overall assessment of multicollinearity

---

14.6.7 Addressing Multicollinearity

If multicollinearity is detected, consider the following solutions:

1. **Remove or combine correlated variables:** Drop one of the correlated predictors or create an index/aggregate.
2. **Principal Component Analysis:** Reduce dimensionality by transforming correlated variables into uncorrelated components.
3. Ridge Regression **(L2 regularization):** Introduces a penalty term to stabilize coefficient estimates in the presence of multicollinearity.
4. **Centering variables:** Mean-centering can help reduce multicollinearity, especially in interaction terms.

Multicollinearity can significantly distort regression estimates, leading to misleading interpretations. While VIF and Tolerance are commonly used diagnostics, advanced techniques like the Condition Index and Eigenvalue Decomposition provide deeper insights, especially when dealing with complex datasets.

```r
# Install and load necessary libraries
# install.packages("car")        # For VIF calculation
# install.packages("corpcor")    # For determinant of correlation matrix


library(car)
library(corpcor)

# Simulated dataset with multicollinearity
set.seed(123)
n <- 100
x1 <- rnorm(n, mean = 50, sd = 10)
x2 <- 0.8 * x1 + rnorm(n, sd = 2)   # Highly correlated with x1
x3 <- rnorm(n, mean = 30, sd = 5)
y <- 5 + 0.4 * x1 - 0.3 * x2 + 0.2 * x3 + rnorm(n)

# Original regression model
model <- lm(y ~ x1 + x2 + x3)

# ----------------------------------------------------------------------
# 1. Variance Inflation Factor (VIF)
# ----------------------------------------------------------------------
# Null Hypothesis: No multicollinearity (VIF = 1)
vif_values <- vif(model)
print(vif_values)
#>        x1        x2        x3
#> 14.969143 14.929013  1.017576

# ----------------------------------------------------------------------
# 2. Tolerance Statistic (Reciprocal of VIF)
# ----------------------------------------------------------------------
tolerance_values <- 1 / vif_values
print(tolerance_values)
#>         x1         x2         x3
#> 0.06680409 0.06698366 0.98272742

# ----------------------------------------------------------------------
# 3. Condition Index and Eigenvalue Decomposition
# ----------------------------------------------------------------------
# Scaling the independent variables
X <- model.matrix(model)[,-1]  # Removing intercept
eigen_decomp <-
    eigen(cor(X))   # Eigenvalue decomposition of the correlation matrix

# Condition Index
```

```r
condition_index <-
    sqrt(max(eigen_decomp$values) / eigen_decomp$values)
print(condition_index)
#> [1] 1.000000 1.435255 7.659566

# Variance Decomposition Proportions (VDP)
# Proportions calculated based on the squared coefficients
loadings <- eigen_decomp$vectors
vdp <- apply(loadings ^ 2, 2, function(x)
    x / sum(x))
print(vdp)
#>            [,1]       [,2]         [,3]
#> [1,] 0.48567837 0.01363318 5.006885e-01
#> [2,] 0.48436754 0.01638399 4.992485e-01
#> [3,] 0.02995409 0.96998283 6.307954e-05

# ----------------------------------------------------------------------
# 4. Pairwise Correlation Matrix
# ----------------------------------------------------------------------
correlation_matrix <- cor(X)
print(correlation_matrix)
#>           x1         x2         x3
#> x1  1.000000  0.9659070 -0.1291760
#> x2  0.965907  1.0000000 -0.1185042
#> x3 -0.129176 -0.1185042  1.0000000

# ----------------------------------------------------------------------
# 5. Determinant of the Correlation Matrix
# ----------------------------------------------------------------------
determinant_corr_matrix <- det(correlation_matrix)
cat("Determinant of the Correlation Matrix:",
    determinant_corr_matrix,
    "\n")
#> Determinant of the Correlation Matrix: 0.06586594
```

Interpretation of the Results

1. **Variance Inflation Factor (VIF)**
   - **Formula:** $\text{VIF}_j = \frac{1}{1-R_j^2}$
   - **Decision Rule:**
     - **VIF** ≈ 1: No multicollinearity.
     - $1 < \text{VIF} < 5$: Moderate correlation, usually acceptable.
     - $\text{VIF} \geq 5$: High correlation; investigate further.
     - $\text{VIF} \geq 10$: Severe multicollinearity; corrective action recom-

mended.

2. **Tolerance Statistic**
   - **Formula:** $\text{Tolerance}_j = 1 - R_j^2 = \frac{1}{\text{VIF}_j}$
   - **Decision Rule:**
     - **Tolerance** > 0.2: Low risk of multicollinearity.
     - **Tolerance** < 0.2: Possible multicollinearity problem.
     - **Tolerance** < 0.1: Severe multicollinearity.
3. **Condition Index and Eigenvalue Decomposition**
   - **Formula:** $\text{CI}_j = \sqrt{\frac{\lambda_{\max}}{\lambda_j}}$
   - **Decision Rule:**
     - **CI** < 10: No significant multicollinearity.
     - $10 \leq \text{CI} < 30$: Moderate to strong multicollinearity.
     - **CI** ≥ 30: Severe multicollinearity.
   - **Variance Decomposition Proportions (VDP):**
     - High VDP (> 0.5) associated with high CI indicates problematic variables.
4. **Pairwise Correlation Matrix**
   - **Decision Rule:**
     - $|\rho| < 0.5$: Weak correlation.
     - $0.5 \leq |\rho| < 0.8$: Moderate correlation; monitor.
     - $|\rho| \geq 0.8$: Strong correlation; potential multicollinearity issue.
5. **Determinant of the Correlation Matrix**
   - **Decision Rule:**
     - $\det(R) \approx 1$: No multicollinearity.
     - $\det(R) \approx 0$: Severe multicollinearity (near-singular matrix).

Model specification tests are essential for diagnosing and validating econometric models. They ensure that the model assumptions hold true, thereby improving the accuracy and reliability of the estimations. By systematically applying these tests, researchers can identify issues related to nested and non-nested models, heteroskedasticity, functional form, endogeneity, autocorrelation, and multicollinearity, leading to more robust and credible econometric analyses.

Chapter 15

Variable Selection

Imagine you're a detective standing before a pinboard covered in clues—some are glaringly obvious, while others might be red herrings. Your mission? To pick which pieces of evidence will crack the case. This is the essence of variable selection in statistics: deciding which variables best uncover the story behind your data. Far from a mechanical chore, it's a high-stakes balancing act blending analytical goals, domain insights, data realities, and computational feasibility.

Why Does Variable Selection Matter?

- **Focus and Clarity**: Models cluttered with unnecessary variables can obscure the real relationships or patterns in your data. By identifying the variables that truly drive your results, you sharpen your model's focus and interpretability.
- **Efficiency and Performance**: Too many variables can lead to overfitting—fitting the quirks of a single dataset rather than underlying trends. Streamlined models often run faster and generalize better.
- **Practical Constraints**: In many real-world scenarios, data collection or processing costs money, time, and effort. Prioritizing the most meaningful variables becomes not just a statistical concern, but a strategic one.

Key Influences on Variable Selection

1. **Objectives or Goals**
    - *Prediction vs. Inference*: Are you trying to forecast future outcomes or explain why certain events happen? Prediction-focused models might include as many relevant features as possible for accuracy, whereas inference-driven models often strive for parsimony and clearer relationships.

- *Balance*: Some analyses blend both objectives, requiring careful negotiation between complexity (to maximize predictive ability) and simplicity (to maintain interpretability).

2. **Previously Acquired Expertise**
   - *Domain Knowledge*: Whether you're analyzing financial trends or studying medical records, familiarity with the subject can reveal which variables are naturally linked to the phenomenon.
   - *Subtle Clues*: Experts can uncover hidden confounders—variables that outwardly seem irrelevant yet dramatically influence results.
3. **Availability and Quality of Data**
   - *Completeness*: Missing data or sparse measurements can force you to discard or transform variables. Sometimes the ideal variable simply isn't present in your dataset.
   - *Reliability*: A variable riddled with measurement errors or inconsistencies may do more harm than good.
4. **Computational Resources and Software**
   - *Toolset Capabilities*: Some statistical techniques or advanced machine learning methods thrive on large sets of variables, while others become unwieldy.
   - *Time and Memory Constraints*: Even the most sophisticated algorithms can choke on too much data if hardware resources are limited.

Selecting the right subset of variables enhances model interpretability, reduces computational cost, and prevents overfitting. Broadly, variable selection methods fall into three categories:

1. **Filter Methods**: Use statistical properties of the data to select features before modeling.
   1. Information Criteria-Based Selection
      1. Akaike Information Criterion (AIC)
      2. Bayesian Information Criterion (BIC)
      3. Mallows's C Statistic
      4. Hannan-Quinn Criterion (HQC)
      5. Minimum Description Length (MDL)
      6. Adjusted R^2
      7. Prediction Error Sum of Squares (PRESS)
   2. Univariate Selection Methods
   3. Correlation-Based Feature Selection
   4. Variance Thresholding
2. **Wrapper Methods**: Evaluate different subsets of features based on model performance.
   1. Exhaustive Search (Best Subsets Algorithm)

   2. Best Subsets Algorithm
   3. Stepwise Selection Methods
      1. Forward Selection
      2. Backward Elimination
      3. Stepwise (Both Directions) Selection
   4. Branch-and-Bound Algorithm
   5. Recursive Feature Elimination
3. **Embedded Methods**: Perform feature selection as part of the model training process.
   1. Lasso Regression (L1 Regularization)
   2. Ridge Regression (L2 Regularization)
   3. Elastic Net (Combination of L1 and L2)
   4. Tree-Based Feature Importance
   5. Genetic Algorithms

Table 15.1: Throughout this chapter, let P denote the number of potential predictor variables $(X_1, X_2, \dots, X_{P-1})$.

Method Category	Examples	Pros	Cons
Filter	AIC, BIC, Mutual Info, CFS	Fast, scalable, model-agnostic	Ignores feature interactions
Wrapper	Stepwise Selection, RFE	Finds optimal feature subsets	Computationally expensive
Embedded	Lasso, Decision Trees	Model-aware, balances efficiency	Selection tied to specific models

---

15.1 Filter Methods (Statistical Criteria, Model-Agnostic)

15.1.1 Information Criteria-Based Selection

15.1.1.1 Mallows's C Statistic

The C_p statistic (Mallows, 1973, *Technometrics*, 15, 661-675) (Mallows, 1995) is a criterion used to evaluate the predictive ability of a fitted model. It balances model complexity and goodness-of-fit.

For a model with p parameters, let $\hat{Y}_{ip}$ be the predicted value of Y_i. The **total standardized mean square error of prediction** is:

$$\Gamma_p = \frac{\sum_{i=1}^{n} E(\hat{Y}_{ip} - E(Y_i))^2}{\sigma^2}$$

Expanding Γ_p:

$$\Gamma_p = \frac{\sum_{i=1}^n [E(\hat{Y}_{ip}) - E(Y_i)]^2 + \sum_{i=1}^n \text{Var}(\hat{Y}_{ip})}{\sigma^2}$$

- The **first term** in the numerator represents the squared bias.
- The **second term** represents the prediction variance.

Key Insights

1. **Bias-Variance Tradeoff**:
   - The **bias** decreases as more variables are added to the model.
   - If the **full model** ($p = P$) is assumed to be the true model, $E(\hat{Y}_{ip}) - E(Y_i) = 0$, implying no bias.
   - The **prediction variance** increases as more variables are added: $\sum \text{Var}(\hat{Y}_{ip}) = p\sigma^2$.
   - Therefore, the optimal model balances bias and variance by minimizing Γ_p.
2. **Estimating Γ_p**: Since Γ_p depends on unknown parameters (e.g., β), we use an estimate:

$$C_p = \frac{SSE_p}{\hat{\sigma}^2} - (n - 2p)$$

   - SSE_p: Sum of squared errors for the model with p predictors.
   - $\hat{\sigma}^2$: Mean squared error (MSE) of the full model with all $P - 1$ predictors.
3. **Properties of C_p**:
   - As more variables are added, SSE_p decreases, but the penalty term $2p$ increases.
   - When there is no bias, $E(C_p) \approx p$. Hence, good models have C_p values close to p.
4. **Model Selection Criteria**:
   - **Prediction-focused models**: Consider models with $C_p \leq p$.
   - **Parameter estimation-focused models**: Consider models with $C_p \leq 2p - (P - 1)$ to avoid excess bias.

```r
# Simulated data
set.seed(123)
n <- 100
x1 <- rnorm(n)
x2 <- rnorm(n)
x3 <- rnorm(n)
```

```r
y <- 5 + 3*x1 - 2*x2 + rnorm(n, sd=2)

# Full model and candidate models
full_model <- lm(y ~ x1 + x2 + x3)
model_1 <- lm(y ~ x1)
model_2 <- lm(y ~ x1 + x2)

# Extract SSE and calculate Cp
calculate_cp <- function(model, full_model_sse, full_model_mse, n) {
  sse <- sum(residuals(model)^2)
  p <- length(coefficients(model))
  cp <- (sse / full_model_mse) - (n - 2 * p)
  return(cp)
}

# Full model statistics
full_model_sse <- sum(residuals(full_model)^2)
full_model_mse <- mean(residuals(full_model)^2)

# Cp values for each model
cp_1 <- calculate_cp(model_1, full_model_sse, full_model_mse, n)
cp_2 <- calculate_cp(model_2, full_model_sse, full_model_mse, n)

# Display results
cat("C_p values:\n")
#> C_p values:
cat("Model 1 (y ~ x1):", round(cp_1, 2), "\n")
#> Model 1 (y ~ x1): 83.64
cat("Model 2 (y ~ x1 + x2):", round(cp_2, 2), "\n")
#> Model 2 (y ~ x1 + x2): 6.27
```

For Mallows's C_p criterion, **lower values** are preferred. Specifically:

1. **Ideal Value**: When the model is a good fit and has the correct number of predictors, C_p should be close to the number of predictors p plus 1 (i.e., $p+1$).

2. **Model Comparison**: Among competing models, you generally prefer the one with the smallest C_p, as long as it is close to $p+1$.

3. **Overfitting Indicator**: If C_p is significantly lower than $p+1$, it may suggest overfitting.

4. **Underfitting Indicator**: If C_p is much higher than $p+1$, it suggests the model is underfitting the data and missing important predictors.

15.1.1.2 Akaike Information Criterion (AIC)

The **Akaike Information Criterion (AIC)** is a widely used model selection metric that evaluates the tradeoff between model fit and complexity. It was introduced by Hirotugu Akaike and is rooted in information theory, measuring the relative quality of statistical models for a given dataset.

For a model with p parameters, the AIC is given by:

$$AIC = n \ln \left(\frac{SSE_p}{n} \right) + 2p$$

Where:

- n is the number of observations.
- SSE_p is the sum of squared errors for a model with p parameters.

Key Insights

1. **Components of AIC**:
   - The **first term** ($n \ln(SSE_p/n)$): Reflects the goodness-of-fit of the model. It decreases as SSE_p decreases, meaning the model better explains the data.
   - The **second term** ($2p$): Represents a penalty for model complexity. It increases with the number of parameters to discourage overfitting.
2. **Model Selection Principle**:
   - Smaller AIC values indicate a better balance between fit and complexity.
   - Adding parameters generally reduces SSE_p, but increases the penalty term ($2p$). If AIC increases when a parameter is added, that parameter is likely unnecessary.
3. **Tradeoff**:
   - AIC emphasizes the tradeoff between:
     - **Precision of fit**: Reducing the error in explaining the data.
     - **Parsimony**: Avoiding unnecessary parameters to maintain model simplicity.
4. **Comparative Criterion**:
   - AIC does not provide an absolute measure of model quality; instead, it compares relative performance. The model with the lowest AIC is preferred.

```r
# Simulated data
set.seed(123)
n <- 100
x1 <- rnorm(n)
x2 <- rnorm(n)
```

```r
x3 <- rnorm(n)
y <- 5 + 3*x1 - 2*x2 + rnorm(n, sd=2)

# Candidate models
model_1 <- lm(y ~ x1)
model_2 <- lm(y ~ x1 + x2)
model_3 <- lm(y ~ x1 + x2 + x3)

# Function to manually compute AIC
calculate_aic <- function(model, n) {
  sse <- sum(residuals(model)^2)
  p <- length(coefficients(model))
  aic <- n * log(sse / n) + 2 * p
  return(aic)
}

# Calculate AIC for each model
aic_1 <- calculate_aic(model_1, n)
aic_2 <- calculate_aic(model_2, n)
aic_3 <- calculate_aic(model_3, n)

# Display results
cat("AIC values:\n")
#> AIC values:
cat("Model 1 (y ~ x1):", round(aic_1, 2), "\n")
#> Model 1 (y ~ x1): 207.17
cat("Model 2 (y ~ x1 + x2):", round(aic_2, 2), "\n")
#> Model 2 (y ~ x1 + x2): 150.87
cat("Model 3 (y ~ x1 + x2 + x3):", round(aic_3, 2), "\n")
#> Model 3 (y ~ x1 + x2 + x3): 152.59
```

Interpretation

- Compare the AIC values across models:
  - A smaller AIC indicates a model with a better balance between fit and complexity.
  - If the AIC increases when moving from one model to another (e.g., from Model 2 to Model 3), the additional parameter(s) in the larger model may not be justified.

Advantages:

- Simple to compute and widely applicable.
- Penalizes overfitting, encouraging parsimonious models.

Limitations:

- Assumes the model errors are normally distributed and independent.
- Does not evaluate absolute model fit, only relative performance.
- Sensitive to sample size; for smaller samples, consider using the corrected version, AICc.

Corrected AIC (AICc)

For small sample sizes ($n/p \leq 40$), the corrected AIC, $AICc$, adjusts for the sample size:

$$AICc = AIC + \frac{2p(p+1)}{n-p-1}$$

This adjustment prevents over-penalizing models with more parameters when n is small.

15.1.1.3 Bayesian Information Criterion (BIC)

The **Bayesian Information Criterion (BIC)**, also known as Schwarz Criterion, is another popular metric for model selection. It extends the concept of AIC by introducing a stronger penalty for model complexity, particularly when the number of observations is large. BIC is grounded in Bayesian probability theory and provides a framework for selecting the most plausible model among a set of candidates.

For a model with p parameters, the BIC is defined as:

$$BIC = n \ln\left(\frac{SSE_p}{n}\right) + p\ln(n)$$

Where:

- n is the number of observations.
- SSE_p is the sum of squared errors for the model with p parameters.
- p is the number of parameters, including the intercept.

Key Insights

1. **Components of BIC**:
   - The **first term** ($n\ln(SSE_p/n)$): Measures the goodness-of-fit, similar to AIC. It decreases as SSE_p decreases, indicating a better fit to the data.
   - The **second term** ($p\ln(n)$): Penalizes model complexity. Unlike AIC's penalty ($2p$), the penalty in BIC increases with $\ln(n)$, making it more sensitive to the number of observations.
2. **Model Selection Principle**:
   - Smaller BIC values indicate a better model.

- Adding parameters reduces SSE_p, but the penalty term $p\ln(n)$ grows more rapidly than AIC's $2p$ for large n. This makes BIC more conservative than AIC in selecting models with additional parameters.

3. **Tradeoff**:
   - Like AIC, BIC balances:
     - **Precision of fit**: Capturing the underlying structure in the data.
     - **Parsimony**: Avoiding overfitting by discouraging unnecessary parameters.
   - BIC tends to favor simpler models compared to AIC, particularly when n is large.
4. **Comparative Criterion**:
   - BIC, like AIC, is used to compare models. The model with the smallest BIC is preferred.

```r
# Function to manually compute BIC
calculate_bic <- function(model, n) {
  sse <- sum(residuals(model)^2)
  p <- length(coefficients(model))
  bic <- n * log(sse / n) + p * log(n)
  return(bic)
}

# Calculate BIC for each model
bic_1 <- calculate_bic(model_1, n)
bic_2 <- calculate_bic(model_2, n)
bic_3 <- calculate_bic(model_3, n)

# Display results
cat("BIC values:\n")
#> BIC values:
cat("Model 1 (y ~ x1):", round(bic_1, 2), "\n")
#> Model 1 (y ~ x1): 212.38
cat("Model 2 (y ~ x1 + x2):", round(bic_2, 2), "\n")
#> Model 2 (y ~ x1 + x2): 158.68
cat("Model 3 (y ~ x1 + x2 + x3):", round(bic_3, 2), "\n")
#> Model 3 (y ~ x1 + x2 + x3): 163.01
```

Interpretation

- Compare the BIC values across models:
  - Smaller BIC values suggest a better model.
  - If BIC increases when moving to a larger model, the added complexity may not justify the reduction in SSE_p.

Comparison with AIC

Criterion	Penalty Term	Sensitivity to Sample Size	Preferred Model Selection
AIC	$2p$	Less sensitive	More parameters
BIC	$p\ln(n)$	More sensitive	Simpler models

- BIC generally prefers simpler models than AIC, especially when n is large.
- In small datasets, AIC may perform better because BIC's penalty grows significantly with $\ln(n)$.

Advantages:

- Strong penalty for complexity makes it robust against overfitting.
- Incorporates sample size explicitly, favoring simpler models as n grows.
- Easy to compute and interpret.

Limitations:

- Assumes model errors are normally distributed and independent.
- May underfit in smaller datasets where the penalty term dominates.
- Like AIC, BIC is not an absolute measure of model quality but a relative one.

15.1.1.4 Hannan-Quinn Criterion (HQC)

The **Hannan-Quinn Criterion (HQC)** is a statistical metric for model selection, similar to AIC and BIC. It evaluates the tradeoff between model fit and complexity, offering a middle ground between the conservative penalty of BIC and the less stringent penalty of AIC. HQC is especially useful in time-series modeling and situations where large datasets are involved.

The HQC for a model with p parameters is defined as:

$$HQC = n\ln\left(\frac{SSE_p}{n}\right) + 2p\ln(\ln(n))$$

Where:

- n: Number of observations.
- SSE_p: Sum of Squared Errors for the model with p predictors.
- p: Number of parameters, including the intercept.

Key Insights

1. **Components**:
   - The **first term** ($n \ln(SSE_p/n)$): Measures the goodness-of-fit, similar to AIC and BIC. Smaller SSE indicates a better fit.
   - The **second term** ($2p \ln(\ln(n))$): Penalizes model complexity. The penalty grows logarithmically with the sample size, similar to BIC but less severe.
2. **Model Selection Principle**:
   - Smaller HQC values indicate a better balance between model fit and complexity.
   - Models with lower HQC are preferred.
3. **Penalty Comparison**:
   - HQC's penalty lies between that of AIC and BIC:
     - AIC: $2p$ (less conservative, favors complex models).
     - BIC: $p \ln(n)$ (more conservative, favors simpler models).
     - HQC: $2p \ln(\ln(n))$ (balances AIC and BIC).
4. **Use Case**:
   - HQC is particularly suited for large datasets or time-series models where overfitting is a concern, but BIC may overly penalize model complexity.

```r
# Simulated data
set.seed(123)
n <- 100
x1 <- rnorm(n)
x2 <- rnorm(n)
x3 <- rnorm(n)
x4 <- rnorm(n)
y <- 5 + 3*x1 - 2*x2 + x3 + rnorm(n, sd=2)

# Prepare models
data <- data.frame(y, x1, x2, x3, x4)
model_1 <- lm(y ~ x1, data = data)
model_2 <- lm(y ~ x1 + x2, data = data)
model_3 <- lm(y ~ x1 + x2 + x3, data = data)

# Function to calculate HQC
calculate_hqc <- function(model, n) {
  sse <- sum(residuals(model)^2)
  p <- length(coefficients(model))
  hqc <- n * log(sse / n) + 2 * p * log(log(n))
  return(hqc)
}
```

```r
# Calculate HQC for each model
hqc_1 <- calculate_hqc(model_1, n)
hqc_2 <- calculate_hqc(model_2, n)
hqc_3 <- calculate_hqc(model_3, n)

# Display results
cat("HQC values:\n")
#> HQC values:
cat("Model 1 (y ~ x1):", round(hqc_1, 2), "\n")
#> Model 1 (y ~ x1): 226.86
cat("Model 2 (y ~ x1 + x2):", round(hqc_2, 2), "\n")
#> Model 2 (y ~ x1 + x2): 156.44
cat("Model 3 (y ~ x1 + x2 + x3):", round(hqc_3, 2), "\n")
#> Model 3 (y ~ x1 + x2 + x3): 141.62
```

Interpretation

1. **Comparing HQC Values**:
   - Smaller HQC values indicate a better balance between goodness-of-fit and parsimony.
   - Select the model with the smallest HQC.
2. **Tradeoffs**:
   - HQC balances fit and complexity more conservatively than AIC but less so than BIC.
   - It is particularly useful when overfitting is a concern but avoiding overly simplistic models is also important.

Comparison with Other Criteria

Criterion	Penalty Term	Sensitivity to Sample Size	Preferred Model Selection
AIC	$2p$	Low	Favors more complex models
BIC	$p \ln(n)$	High	Favors simpler models
HQC	$2p \ln(\ln(n))$	Moderate	Balances fit and parsimony

Advantages:

- Less sensitive to sample size than BIC, avoiding excessive penalization for large datasets.

- Provides a balanced approach to model selection, reducing the risk of overfitting while avoiding overly simplistic models.
- Particularly useful in time-series analysis.

Limitations:

- Like AIC and BIC, assumes model errors are normally distributed and independent.
- HQC is not as widely implemented in statistical software, requiring custom calculations.

Practical Considerations

- **When to use HQC?**
  - When both AIC and BIC provide conflicting recommendations.
  - For large datasets or time-series models where BIC may overly penalize complexity.
- **When to use AIC or BIC?**
  - AIC for smaller datasets or when the goal is prediction.
  - BIC for large datasets or when parsimony is critical.

15.1.1.5 Minimum Description Length (MDL)

The **Minimum Description Length (MDL)** principle is a model selection framework rooted in information theory. It balances model complexity and goodness-of-fit by seeking the model that minimizes the total length of encoding the data and the model itself. MDL is a generalization of other model selection criteria like AIC and BIC but offers a more theoretical foundation.

Theoretical Foundation

MDL is based on the idea that the best model is the one that compresses the data most efficiently. It represents a tradeoff between:

1. **Model Complexity**: The cost of describing the model, including the number of parameters.
2. **Data Fit**: The cost of describing the data given the model.

The total description length is expressed as:

$$L(M, D) = L(M) + L(D|M)$$

Where:

- $L(M)$: The length of encoding the model (complexity of the model).

- $L(D|M)$: The length of encoding the data given the model (fit to the data).

Key Insights

1. **Model Complexity ($L(M)$)**:
   - More complex models require longer descriptions, as they involve more parameters.
   - Simpler models are favored unless the added complexity significantly improves the fit.
2. **Data Fit ($L(D|M)$)**:
   - Measures how well the model explains the data.
   - Poorly fitting models require more bits to describe the residual error.
3. **Tradeoff**:
   - MDL balances these two components, selecting the model that minimizes the total description length.

Connection to Other Criteria

- MDL is closely related to BIC. In fact, the BIC criterion can be derived as an approximation of MDL for certain statistical models:

$$BIC = n \ln(SSE_p/n) + p \ln(n)$$

- However, MDL is more flexible and does not rely on specific assumptions about the error distribution.

Practical Use Cases

- **Time-Series Modeling**: MDL is particularly effective for selecting models in time-series data, where overfitting is common.
- **Machine Learning**: MDL is used in regularization techniques and decision tree pruning to prevent overfitting.
- **Signal Processing**: In applications such as compression and coding, MDL directly guides optimal model selection.

```r
# Simulated data
set.seed(123)
n <- 100
x1 <- rnorm(n)
x2 <- rnorm(n)
x3 <- rnorm(n)
y <- 5 + 3*x1 - 2*x2 + x3 + rnorm(n, sd=2)
```

```r
# Prepare models
data <- data.frame(y, x1, x2, x3)
model_1 <- lm(y ~ x1, data = data)
model_2 <- lm(y ~ x1 + x2, data = data)
model_3 <- lm(y ~ x1 + x2 + x3, data = data)

# Function to calculate MDL
calculate_mdl <- function(model, n) {
  sse <- sum(residuals(model)^2)
  p <- length(coefficients(model))
  mdl <- p * log(n) + n * log(sse / n)
  return(mdl)
}

# Calculate MDL for each model
mdl_1 <- calculate_mdl(model_1, n)
mdl_2 <- calculate_mdl(model_2, n)
mdl_3 <- calculate_mdl(model_3, n)

# Display results
cat("MDL values:\n")
#> MDL values:
cat("Model 1 (y ~ x1):", round(mdl_1, 2), "\n")
#> Model 1 (y ~ x1): 219.87
cat("Model 2 (y ~ x1 + x2):", round(mdl_2, 2), "\n")
#> Model 2 (y ~ x1 + x2): 173.42
cat("Model 3 (y ~ x1 + x2 + x3):", round(mdl_3, 2), "\n")
#> Model 3 (y ~ x1 + x2 + x3): 163.01
```

Interpretation

1. **Choosing the Best Model**:
   - The model with the smallest MDL value is preferred, as it achieves the best tradeoff between fit and complexity.
2. **Practical Implications**:
   - MDL discourages overfitting by penalizing complex models that do not significantly improve data fit.

Advantages:

- Theoretically grounded in information theory.
- Offers a natural framework for balancing complexity and fit.
- Flexible and can be applied across various modeling frameworks.

Limitations:

- Computationally intensive, especially for non-linear models.
- Requires careful formulation of $L(M)$ and $L(D | M)$ for non-standard models.
- Less common in standard statistical software compared to AIC or BIC.

15.1.1.6 Prediction Error Sum of Squares (PRESS)

The **Prediction Error Sum of Squares (PRESS)** statistic measures the predictive ability of a model by evaluating how well it performs on data not used in fitting the model. PRESS is particularly useful for assessing model validity and identifying overfitting.

The PRESS statistic for a model with p parameters is defined as:

$$PRESS_p = \sum_{i=1}^{n} (Y_i - \hat{Y}_{i(i)})^2$$

Where:

- $\hat{Y}_{i(i)}$ is the prediction of the i-th response when the i-th observation is **omitted** during model fitting.
- Y_i is the observed response for the i-th observation.

Key Insights

1. **Leave-One-Out Cross-Validation (LOOCV)**:
   - PRESS is computed by excluding each observation one at a time and predicting its response using the remaining data.
   - This process evaluates the model's generalizability and reduces overfitting.
2. **Model Selection Principle**:
   - Smaller values of $PRESS_p$ indicate better predictive performance.
   - A small $PRESS_p$ suggests that the model captures the underlying structure of the data without overfitting.
3. **Computational Complexity**:
   - Computing $\hat{Y}_{i(i)}$ for each observation can be computationally intensive for models with large p or datasets with many observations.
   - Alternative approximations (e.g., using leverage values) can simplify computations.

```r
# Function to compute PRESS
calculate_press <- function(model) {
  residuals <- residuals(model)
  h <- lm.influence(model)$hat # leverage values
  press <- sum((residuals / (1 - h))^2) # PRESS formula using leverage
  return(press)
}

# Calculate PRESS for each model
press_1 <- calculate_press(model_1)
press_2 <- calculate_press(model_2)
press_3 <- calculate_press(model_3)

# Display results
cat("PRESS values:\n")
#> PRESS values:
cat("Model 1 (y ~ x1):", round(press_1, 2), "\n")
#> Model 1 (y ~ x1): 854.36
cat("Model 2 (y ~ x1 + x2):", round(press_2, 2), "\n")
#> Model 2 (y ~ x1 + x2): 524.56
cat("Model 3 (y ~ x1 + x2 + x3):", round(press_3, 2), "\n")
#> Model 3 (y ~ x1 + x2 + x3): 460
```

Interpretation

- Compare the PRESS values across models:
  - Models with smaller PRESS values are preferred as they exhibit better predictive ability.
  - A large PRESS value indicates potential overfitting or poor model generalizability.

Advantages:

- Provides an unbiased measure of predictive performance.
- Helps identify overfitting by simulating the model's performance on unseen data.

Limitations:

- Computationally intensive for large datasets or models with many predictors.
- Sensitive to influential observations; high-leverage points can disproportionately affect results.

Alternative Approaches

To address the computational challenges of PRESS, alternative methods can be employed:

- **Approximation using leverage values**: As shown in the example, leverage values simplify the calculation of $\widehat{Y}_{i(i)}$.
- **K-Fold Cross-Validation**: Dividing the dataset into k folds reduces computational burden compared to LOOCV while still providing robust estimates.

15.1.2 Univariate Selection Methods

Univariate selection methods evaluate individual variables in isolation to determine their relationship with the target variable. These methods are often categorized under filter methods, as they do not involve any predictive model but instead rely on statistical significance and information-theoretic measures.

Univariate selection is particularly useful for:

- **Preprocessing large datasets** by eliminating irrelevant features.
- **Reducing dimensionality** before applying more complex feature selection techniques.
- **Improving interpretability** by identifying the most relevant features.

The two main categories of univariate selection methods are:

1. **Statistical Tests**: Evaluate the significance of relationships between individual features and the target variable.
2. **Information-Theoretic Measures**: Assess the dependency between variables based on information gain and mutual information.

15.1.2.1 Statistical Tests

Statistical tests assess the significance of relationships between individual predictors and the target variable. The choice of test depends on the type of data:

Test	Used For	Example Use Case
Chi-Square Test	Categorical predictors vs. Categorical target	Checking if gender affects purchase behavior
ANOVA (Analysis of Variance)	Continuous predictors vs. Categorical target	Testing if different income groups have varying spending habits

Test	Used For	Example Use Case
Correlation Coefficients	Continuous predictors vs. Continuous target	Measuring the relationship between advertising spend and sales

Check out Descriptive Statistics and Basic Statistical Inference for more details.

15.1.2.2 Information-Theoretic Measures

Information-theoretic measures assess variable relevance based on how much information they provide about the target.

15.1.2.2.1 Information Gain Information Gain (IG) measures the reduction in uncertainty about the target variable when a predictor is known. It is used extensively in decision trees.

Formula:

$$IG = H(Y) - H(Y|X)$$

Where:

- $H(Y)$ = Entropy of target variable
- $H(Y|X)$ = Conditional entropy of target given predictor

A higher IG indicates a more informative variable.

```
# Load Library
library(FSelector)

# Example: Computing Information Gain
data(iris)
information.gain(Species ~ ., iris)
#>              attr_importance
#> Sepal.Length       0.4521286
#> Sepal.Width        0.2672750
#> Petal.Length       0.9402853
#> Petal.Width        0.9554360
```

15.1.2.2.2 Mutual Information Mutual Information (MI) quantifies how much knowing one variable reduces uncertainty about another. Unlike correlation, it captures both linear and non-linear relationships.

- **Pros**: Captures non-linear relationships, robust to outliers.
- **Cons**: More computationally intensive than correlation.

Formula:

$$MI(X,Y) = \sum_{x,y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)}$$

Where:

- $P(x,y)$ = Joint probability distribution of X and Y.
- $P(x)$, $P(y)$ = Marginal probability distributions.

```r
# Load Library
library(infotheo)

# Compute Mutual Information Between Two Features
set.seed(123)
X <- sample(1:5, 100, replace=TRUE)
Y <- sample(1:5, 100, replace=TRUE)

mutinformation(X, Y)
#> [1] 0.06852247
```

- Since `X` and `Y` are independently sampled, we expect them to have no mutual dependence.
- The MI value should be close to 0, indicating that knowing `X` provides almost no information about `Y`, and vice versa.
- If the MI value is significantly greater than 0, it could be due to random fluctuations, especially in small samples.

Method	Type	Suitable For	Pros	Cons
Chi-Square Test	Statistical Test	Categorical vs. Categorical	Simple, interpretable	Requires large sample sizes
ANOVA	Statistical Test	Continuous vs. Categorical	Handles multiple groups	Assumes normality
Correlation	Statistical Test	Continuous vs. Continuous	Easy to compute	Only captures linear relations
Information Gain	Information-Based	Any Variable Type	Good for decision trees	Requires computation of entropy
Mutual Information	Information-Based	Any Variable Type	Captures non-linear dependencies	More computationally expensive

15.1.3 Correlation-Based Feature Selection

Evaluates features based on their correlation with the target and redundancy with other features. Check out Descriptive Statistics and Basic Statistical Inference for more details.

15.1.4 Variance Thresholding

Variance Thresholding is a simple yet effective filter method used for feature selection. It removes features with **low variance**, assuming that low-variance features contribute little to model prediction. This technique is particularly useful when:

- **Handling high-dimensional datasets** where many features contain little useful information.
- **Reducing computational complexity** by removing uninformative features.
- **Avoiding overfitting** by eliminating features that are nearly constant across samples.

This method is most effective when dealing with **binary features** (e.g., categorical variables encoded as 0s and 1s) and **numerical features** with low variance.

Variance measures the **spread** of a feature's values. A feature with **low variance** contains nearly the same value for all observations, making it less useful for predictive modeling.

For a feature X, variance is calculated as:

$$Var(X) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2$$

where:

- X_i is an individual observation,
- $\bar{X}$ is the mean of X,
- n is the number of observations.

Example: Features with Low and High Variance

Feature	Sample Values	Variance
Low Variance	0, 0, 0, 1, 0, 0, 0, 0, 0, 0	0.04
High Variance	3, 7, 1, 9, 2, 8, 6, 4, 5, 0	7.25

15.1.4.1 Identifying Low-Variance Features

A **variance threshold** is set to remove features below a certain variance level. The default threshold is **0**, which removes features with a single constant value across all samples.

```
# Load necessary library
library(caret)

# Generate synthetic dataset
set.seed(123)
data <- data.frame(
  Feature1 = c(rep(0, 50), rep(1, 50)), # Low variance
  Feature2 = rnorm(100, mean=10, sd=1), # High variance
  Feature3 = runif(100, min=5, max=15), # Moderate variance
  Feature4 = c(rep(3, 95), rep(4, 5))  # Almost constant
)


# Compute Variance of Features
variances <- apply(data, 2, stats::var)
print(variances)
#>  Feature1  Feature2  Feature3  Feature4
#> 0.2525253 0.8332328 8.6631461 0.0479798

# Set threshold and remove low-variance features
threshold <- 0.1
selected_features <- names(variances[variances > threshold])
filtered_data <- data[, selected_features]

print(selected_features)  # Remaining features after filtering
#> [1] "Feature1" "Feature2" "Feature3"
```

15.1.4.2 Handling Binary Categorical Features

For **binary features** (0/1 values), variance is computed as:

$$Var(X) = p(1-p)$$

where p is the proportion of **ones**.

- If $p \approx 0$ or $p \approx 1$, variance is **low**, meaning the feature is almost constant.
- If $p = 0.5$, variance is **highest**, meaning equal distribution of 0s and 1s.

```
# Binary feature dataset
binary_data <- data.frame(
  Feature_A = c(rep(0, 98), rep(1, 2)), # Low variance (almost all 0s)
  Feature_B = sample(0:1, 100, replace=TRUE) # Higher variance
```

```
)

# Compute variance for binary features
binary_variances <- apply(binary_data, 2, stats::var)
print(binary_variances)
#>  Feature_A  Feature_B
#> 0.01979798 0.24757576

# Apply threshold (removing features with variance < 0.01)
threshold <- 0.01
filtered_binary <- binary_data[, binary_variances > threshold]
print(colnames(filtered_binary))
#> [1] "Feature_A" "Feature_B"
```

Aspect	Pros	Cons
Efficiency	Fast and computationally cheap	May remove useful features
Interpretability	Simple to understand and implement	Ignores correlation with target
Applicability	Works well for removing constant or near-constant features	Not useful for detecting redundant but high-variance features

---

15.2 Wrapper Methods (Model-Based Subset Evaluation)

15.2.1 Best Subsets Algorithm

The **Best Subsets Algorithm** is a systematic method for selecting the best combination of predictors. Unlike exhaustive search, which evaluates all possible subsets of predictors, this algorithm efficiently narrows the search space while guaranteeing the identification of the best subset for each size.

- The algorithm is based on the **"leap and bounds"** method introduced by (Furnival and Wilson, 2000).
- It combines:
  - **Comparison of SSE**: Evaluates models by their Sum of Squared Errors.
  - **Control over sequence**: Optimizes the order in which subset models are computed.
- **Guarantees**: Finds the best m subset models within each subset size while reducing computational burden compared to evaluating all possible

subsets.

Key Features

1. **Subset Comparison**:
   - The algorithm ranks subsets based on a criterion such as R^2, adjusted R^2, AIC, or BIC.
   - It evaluates models of varying sizes, starting from 1 predictor to p predictors.
2. **Efficiency**:
   - By leveraging "leap and bounds," the algorithm avoids evaluating subsets unlikely to yield the best results.
   - This reduces the computational cost significantly compared to evaluating 2^p subsets in exhaustive search.
3. **Output**:
   - Produces the best subsets for each model size, which can be compared using criteria like AIC, BIC, or PRESS.

```r
# Load the leaps package
library("leaps")

# Simulated data
set.seed(123)
n <- 100
x1 <- rnorm(n)
x2 <- rnorm(n)
x3 <- rnorm(n)
x4 <- rnorm(n)
y <- 5 + 3*x1 - 2*x2 + x3 + rnorm(n, sd=2)

# Prepare data for best subsets model
data <- data.frame(y, x1, x2, x3, x4)

# Perform best subsets model
best_subsets <- regsubsets(y ~ ., data = data, nvmax = 4)

# Summarize results
best_summary <- summary(best_subsets)

# Display model selection metrics
cat("Best Subsets Summary:\n")
#> Best Subsets Summary:
cat("Adjusted R^2:\n", best_summary$adjr2, "\n")
#> Adjusted R^2:
```

```
#>  0.3651194 0.69237 0.7400424 0.7374724
cat("Cp:\n", best_summary$cp, "\n")
#> Cp:
#>  140.9971 19.66463 3.060205 5
cat("BIC:\n", best_summary$bic, "\n")
#> BIC:
#>  -37.23673 -106.1111 -119.3802 -114.8383

# Visualize results
plot(best_subsets, scale = "adjr2")
title("Best Subsets: Adjusted R^2")
```

Interpretation

1. **Model Size**:
   - Examine the metrics for models with 1, 2, 3, and 4 predictors.
   - Choose the model size that optimizes your preferred metric (e.g., maximized adjusted R^2, minimized BIC).
2. **Model Comparison**:
   - For small datasets, adjusted R^2 is often a reliable criterion.
   - For larger datasets, use BIC or AIC to avoid overfitting.
3. **Efficiency**:
   - The algorithm evaluates far fewer models than an exhaustive search while still guaranteeing optimal results for each subset size.

Advantages:

- Computationally efficient compared to evaluating all possible subsets.
- Guarantees the best subsets for each model size.
- Flexibility to use different selection criteria (e.g., R^2, AIC, BIC).

Limitations:

- May become computationally intensive for very large p (e.g., hundreds of predictors).
- Assumes linear relationships among predictors and the outcome.

Practical Considerations

- **When to use Best Subsets?**
  - For datasets with a moderate number of predictors ($p \leq 20$).
  - When you need an optimal solution for each subset size.
- **Alternatives**:
  - **Stepwise Selection**: Faster but less reliable.
  - **Regularization Techniques**: LASSO and Ridge regression handle large p and collinearity effectively.

15.2.2 Stepwise Selection Methods

Stepwise selection procedures are iterative methods for selecting predictor variables. These techniques balance model simplicity and predictive accuracy by systematically adding or removing variables based on predefined criteria.

Notes:

- Computer implementations often replace exact F-values with "significance" levels:
  - **SLE**: Significance level to enter.
  - **SLS**: Significance level to stay.
- These thresholds serve as guides rather than strict tests of significance.

Balancing SLE and SLS:

- Large **SLE** values: May include too many variables, risking overfitting.
- Small **SLE** values: May exclude important variables, leading to underfitting and overestimation of σ^2.
- A reasonable range for SLE is **0.05 to 0.5**.
- Practical advice:

- If **SLE > SLS**, cycling may occur (adding and removing the same variable repeatedly). To fix this, set $SLS = SLE/2$ (Bendel and Afifi, 1977).
- If **SLE < SLS**, the procedure becomes conservative, retaining variables with minimal contribution.

15.2.2.1 Forward Selection

Forward Selection starts with an empty model (only the intercept) and sequentially adds predictors. At each step, the variable that most improves the model fit (based on criteria like R^2, AIC, or F-statistic) is added. The process stops when no variable improves the model significantly.

Steps

1. Begin with the null model: $Y = \beta_0$.
2. Evaluate each predictor's contribution to the model (e.g., using F-statistic or AIC).
3. Add the predictor with the most significant improvement to the model.
4. Repeat until no remaining variable significantly improves the model.

15.2.2.2 Backward Elimination

Backward Elimination starts with the full model, containing all predictors, and sequentially removes the least significant predictor (based on criteria like p-value or F-statistic). The process stops when all remaining predictors meet the significance threshold.

Steps

1. Begin with the full model: $Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$.
2. Identify the predictor with the smallest contribution to the model (e.g., highest p-value).
3. Remove the least significant predictor.
4. Repeat until all remaining predictors are statistically significant.

15.2.2.3 Stepwise (Both Directions) Selection

Stepwise Selection combines forward selection and backward elimination. At each step, it evaluates whether to add or remove predictors based on predefined criteria. This iterative process ensures that variables added in earlier steps can be removed later if they no longer contribute significantly.

Steps

1. Start with the null model or a user-specified initial model.
2. Evaluate all predictors not in the model for inclusion (forward step).

3. Evaluate all predictors currently in the model for removal (backward step).
4. Repeat steps 2 and 3 until no further addition or removal improves the model.

15.2.2.4 Comparison of Methods

Method	Starting Point	Adds Predictors	Removes Predictors	Pros	Cons
Forward Selection	Null model (no terms)	Yes	No	Simple, fast, useful for small models	Cannot remove irrelevant predictors
Backward Elimination	Full model (all terms)	No	Yes	Removes redundant variables	May exclude important variables early
Stepwise Selection	User-defined or null	Yes	Yes	Combines flexibility of both methods	More computationally intensive

```r
# Simulated Data
set.seed(123)
n <- 100
x1 <- rnorm(n)
x2 <- rnorm(n)
x3 <- rnorm(n)
x4 <- rnorm(n)
y <- 5 + 3 * x1 - 2 * x2 + x3 + rnorm(n, sd = 2)

# Prepare data
data <- data.frame(y, x1, x2, x3, x4)

# Null and Full Models
null_model <- lm(y ~ 1, data = data)
full_model <- lm(y ~ ., data = data)

# Forward Selection
forward_model <- step(
    null_model,
    scope = list(lower = null_model, upper = full_model),
    direction = "forward"
)
```

```
#> Start:  AIC=269.2
#> y ~ 1
#>
#>         Df Sum of Sq     RSS    AIC
#> + x1     1    537.56  909.32 224.75
#> + x2     1    523.27  923.62 226.31
#> <none>               1446.88 269.20
#> + x3     1     23.56 1423.32 269.56
#> + x4     1      8.11 1438.78 270.64
#>
#> Step:  AIC=224.75
#> y ~ x1
#>
#>         Df Sum of Sq    RSS    AIC
#> + x2     1    473.21 436.11 153.27
#> + x3     1     62.65 846.67 219.61
#> <none>               909.32 224.75
#> + x4     1      3.34 905.98 226.38
#>
#> Step:  AIC=153.27
#> y ~ x1 + x2
#>
#>         Df Sum of Sq    RSS    AIC
#> + x3     1    71.382 364.73 137.40
#> <none>               436.11 153.27
#> + x4     1     0.847 435.27 155.08
#>
#> Step:  AIC=137.4
#> y ~ x1 + x2 + x3
#>
#>         Df Sum of Sq    RSS    AIC
#> <none>               364.73 137.40
#> + x4     1     0.231 364.50 139.34

# Backward Elimination
backward_model <- step(full_model, direction = "backward")
#> Start:  AIC=139.34
#> y ~ x1 + x2 + x3 + x4
#>
#>         Df Sum of Sq    RSS    AIC
#> - x4     1      0.23 364.73 137.40
#> <none>               364.50 139.34
#> - x3     1     70.77 435.27 155.08
#> - x2     1    480.14 844.64 221.37
#> - x1     1    525.72 890.22 226.63
```

```
#>
#> Step:  AIC=137.4
#> y ~ x1 + x2 + x3
#>
#>        Df Sum of Sq    RSS    AIC
#> <none>              364.73 137.40
#> - x3    1     71.38 436.11 153.27
#> - x2    1    481.94 846.67 219.61
#> - x1    1    528.02 892.75 224.91

# Stepwise Selection (Both Directions)
stepwise_model <- step(
    null_model,
    scope = list(lower = null_model, upper = full_model),
    direction = "both"
)
#> Start:  AIC=269.2
#> y ~ 1
#>
#>        Df Sum of Sq     RSS    AIC
#> + x1    1    537.56  909.32 224.75
#> + x2    1    523.27  923.62 226.31
#> <none>              1446.88 269.20
#> + x3    1     23.56 1423.32 269.56
#> + x4    1      8.11 1438.78 270.64
#>
#> Step:  AIC=224.75
#> y ~ x1
#>
#>        Df Sum of Sq     RSS    AIC
#> + x2    1    473.21  436.11 153.27
#> + x3    1     62.65  846.67 219.61
#> <none>               909.32 224.75
#> + x4    1      3.34  905.98 226.38
#> - x1    1    537.56 1446.88 269.20
#>
#> Step:  AIC=153.27
#> y ~ x1 + x2
#>
#>        Df Sum of Sq    RSS    AIC
#> + x3    1     71.38 364.73 137.40
#> <none>              436.11 153.27
#> + x4    1      0.85 435.27 155.08
#> - x2    1    473.21 909.32 224.75
#> - x1    1    487.50 923.62 226.31
```

```
#>
#> Step:  AIC=137.4
#> y ~ x1 + x2 + x3
#>
#>         Df Sum of Sq    RSS    AIC
#> <none>               364.73 137.40
#> + x4     1      0.23 364.50 139.34
#> - x3     1     71.38 436.11 153.27
#> - x2     1    481.94 846.67 219.61
#> - x1     1    528.02 892.75 224.91

# Summarize Results
cat("Forward Selection:\n")
#> Forward Selection:
print(summary(forward_model))
#>
#> Call:
#> lm(formula = y ~ x1 + x2 + x3, data = data)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -4.9675 -1.1364  0.1726  1.3983  4.9332
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   5.2332     0.1990  26.300  < 2e-16 ***
#> x1            2.5541     0.2167  11.789  < 2e-16 ***
#> x2           -2.2852     0.2029 -11.263  < 2e-16 ***
#> x3            0.9018     0.2080   4.335  3.6e-05 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.949 on 96 degrees of freedom
#> Multiple R-squared:  0.7479, Adjusted R-squared:   0.74
#> F-statistic: 94.94 on 3 and 96 DF,  p-value: < 2.2e-16

cat("\nBackward Elimination:\n")
#>
#> Backward Elimination:
print(summary(backward_model))
#>
#> Call:
#> lm(formula = y ~ x1 + x2 + x3, data = data)
#>
#> Residuals:
```

```
#>     Min      1Q  Median      3Q     Max
#> -4.9675 -1.1364  0.1726  1.3983  4.9332
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   5.2332     0.1990  26.300  < 2e-16 ***
#> x1            2.5541     0.2167  11.789  < 2e-16 ***
#> x2           -2.2852     0.2029 -11.263  < 2e-16 ***
#> x3            0.9018     0.2080   4.335  3.6e-05 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.949 on 96 degrees of freedom
#> Multiple R-squared:  0.7479, Adjusted R-squared:   0.74
#> F-statistic: 94.94 on 3 and 96 DF,  p-value: < 2.2e-16

cat("\nStepwise Selection:\n")
#>
#> Stepwise Selection:
print(summary(stepwise_model))
#>
#> Call:
#> lm(formula = y ~ x1 + x2 + x3, data = data)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -4.9675 -1.1364  0.1726  1.3983  4.9332
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   5.2332     0.1990  26.300  < 2e-16 ***
#> x1            2.5541     0.2167  11.789  < 2e-16 ***
#> x2           -2.2852     0.2029 -11.263  < 2e-16 ***
#> x3            0.9018     0.2080   4.335  3.6e-05 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.949 on 96 degrees of freedom
#> Multiple R-squared:  0.7479, Adjusted R-squared:   0.74
#> F-statistic: 94.94 on 3 and 96 DF,  p-value: < 2.2e-16
```

Interpretation of Results

1. **Forward Selection**:
    - Starts with no predictors and sequentially adds variables.

- Variables are chosen based on their contribution to improving model fit (e.g., reducing SSE, increasing R^2).

2. **Backward Elimination**:
   - Begins with all predictors and removes the least significant one at each step.
   - Stops when all remaining predictors meet the significance criterion.
3. **Stepwise Selection**:
   - Combines forward selection and backward elimination.
   - At each step, evaluates whether to add or remove variables based on predefined criteria.

Practical Considerations

- **SLE and SLS Tuning**:
  - Choose thresholds carefully to balance model simplicity and predictive performance.
  - For most applications, set $SLS = SLE/2$ to prevent cycling.
- **Order of Entry**:
  - Stepwise selection is unaffected by the order of variable entry. Results depend on the data and significance criteria.
- **Automated Procedures**:
  - Forward selection is simpler but less robust than forward stepwise.
  - Backward elimination works well when starting with all predictors.

Advantages:

- Automates variable selection, reducing manual effort.
- Balances model fit and parsimony using predefined criteria.
- Flexible: Works with different metrics (e.g., SSE, R^2, AIC, BIC).

Limitations:

- Can be sensitive to significance thresholds (SLE, SLS).
- Risk of excluding important variables in datasets with multicollinearity.
- May overfit or underfit if SLE/SLS thresholds are poorly chosen.

Practical Use Cases

- **Forward Stepwise**:
  - When starting with minimal knowledge of predictor importance.

- **Backward Elimination**:
  - When starting with many predictors and needing to reduce model complexity.
- **Stepwise (Both Directions)**:
  - For a balanced approach that adapts as variables are added or removed.

15.2.3 Branch-and-Bound Algorithm

The **Branch-and-Bound Algorithm** is a systematic optimization method used for solving subset selection problems (Furnival and Wilson, 2000). It identifies the best subsets of predictors by exploring the solution space efficiently, avoiding the need to evaluate all possible subsets.

The algorithm is particularly suited for problems with a large number of potential predictors. It systematically evaluates subsets of predictors, using bounds to prune the search space and reduce computational effort.

- **Branching**: Divides the solution space into smaller subsets (branches).
- **Bounding**: Calculates bounds on the best possible solution within each branch to decide whether to explore further or discard.

Key Features

1. **Subset Selection**:
   - Used to identify the best subset of predictors.
   - Evaluates subsets based on criteria like R^2, adjusted R^2, AIC, BIC, or Mallows's C_p.
2. **Efficiency**:
   - Avoids exhaustive search, which evaluates 2^p subsets for p predictors.
   - Reduces the computational burden by eliminating branches that cannot contain the optimal solution.
3. **Guarantee**:
   - Finds the globally optimal subset for the specified criterion.

Algorithm Steps

1. **Initialization**:
   - Start with the full set of predictors.
   - Define the criterion for evaluation (e.g., adjusted R^2, AIC, BIC).
2. **Branching**:
   - Divide the predictors into smaller subsets (branches) systematically.
3. **Bounding**:
   - Compute bounds for the criterion in each branch.

- If the bound indicates the branch cannot improve the current best solution, discard it.

4. **Pruning**:
   - Skip evaluating subsets within discarded branches.
5. **Stopping**:
   - The algorithm terminates when all branches are either evaluated or pruned.

```r
# Load the leaps package
library("leaps")

# Simulated data
set.seed(123)
n <- 100
x1 <- rnorm(n)
x2 <- rnorm(n)
x3 <- rnorm(n)
x4 <- rnorm(n)
y <- 5 + 3 * x1 - 2 * x2 + x3 + rnorm(n, sd = 2)

# Prepare data for subset selection
data <- data.frame(y, x1, x2, x3, x4)

# Perform best subset selection using branch-and-bound
best_subsets <-
    regsubsets(y ~ .,
               data = data,
               nvmax = 4,
               method = "seqrep")

# Summarize results
best_summary <- summary(best_subsets)

# Display results
cat("Best Subsets Summary:\n")
#> Best Subsets Summary:
print(best_summary)
#> Subset selection object
#> Call: regsubsets.formula(y ~ ., data = data, nvmax = 4, method = "seqrep")
#> 4 Variables  (and intercept)
#>    Forced in Forced out
#> x1     FALSE      FALSE
#> x2     FALSE      FALSE
#> x3     FALSE      FALSE
```

```
#> x4     FALSE      FALSE
#> 1 subsets of each size up to 4
#> Selection Algorithm: 'sequential replacement'
#>          x1  x2  x3  x4
#> 1  ( 1 ) "*" " " " " " "
#> 2  ( 1 ) "*" "*" " " " "
#> 3  ( 1 ) "*" "*" "*" " "
#> 4  ( 1 ) "*" "*" "*" "*"

# Visualize best subsets
plot(best_subsets, scale = "adjr2")
title("Best Subsets: AdjusteR^2")
```

Interpretation

1. **Optimal Subsets**:
   - The algorithm identifies the best subset of predictors for each model size.
   - Evaluate models based on metrics like adjusted R^2, BIC, or AIC.
2. **Visualization**:
   - The plot of adjusted R^2 helps identify the model size with the best tradeoff between fit and complexity.
3. **Efficiency**:
   - The Branch-and-Bound Algorithm achieves optimal results without

evaluating all 2^p subsets.

Advantages:

- Guarantees the best subset for a given criterion.
- Reduces computational cost compared to exhaustive search.
- Handles moderate-sized problems efficiently.

Limitations:

- Computationally intensive for very large datasets or many predictors ($p > 20$).
- Requires a clearly defined evaluation criterion.

Practical Considerations

- **When to use Branch-and-Bound?**
  - When p (number of predictors) is moderate ($p \leq 20$).
  - When global optimality for subset selection is essential.
- **Alternatives**:
  - For larger p, consider heuristic methods like stepwise selection or regularization techniques (e.g., LASSO, Ridge).
- **Applications**:
  - Branch-and-Bound is widely used in fields like statistics, operations research, and machine learning where optimal subset selection is crucial.

15.2.4 Recursive Feature Elimination

Recursive Feature Elimination (RFE) is a feature selection method that systematically removes predictors from a model to identify the most relevant subset. RFE is commonly used in machine learning and regression tasks to improve model interpretability and performance by eliminating irrelevant or redundant features.

Theoretical Foundation

1. **Objective**:
   - Select the subset of predictors that maximizes the performance of the model (e.g., minimizes prediction error or maximizes explained variance).
2. **Approach**:

- RFE is a backward selection method that recursively removes the least important predictors based on their contribution to the model's performance.

3. **Key Features**:
   - **Feature Ranking**: Predictors are ranked based on their importance (e.g., weights in a linear model, coefficients in regression, or feature importance scores in tree-based models).
   - **Recursive Elimination**: At each step, the least important predictor is removed, and the model is refit with the remaining predictors.
4. **Evaluation Criterion**:
   - Model performance is evaluated using metrics such as R^2, AIC, BIC, or cross-validation scores.

Steps in RFE

1. **Initialize**:
   - Start with the full set of predictors.
2. **Rank Features**:
   - Train the model and compute feature importance scores (e.g., coefficients, weights, or feature importance).
3. **Eliminate Features**:
   - Remove the least important feature(s) based on the ranking.
4. **Refit Model**:
   - Refit the model with the reduced set of predictors.
5. **Repeat**:
   - Continue the process until the desired number of features is reached.

```r
# Install and load caret package
if (!requireNamespace("caret", quietly = TRUE)) install.packages("caret")
library(caret)

# Simulated data
set.seed(123)
n <- 100
p <- 10
X <- matrix(rnorm(n * p), nrow = n, ncol = p)
y <- 5 + rowSums(X[, 1:3]) + rnorm(n, sd = 2)  # Only first 3 variables are relevant

# Convert to a data frame
data <- as.data.frame(X)
data$y <- y

# Define control for RFE
control <- rfeControl(functions = lmFuncs, # Use linear model for evaluation
```

```
                      method = "cv",       # Use cross-validation
                      number = 10)         # Number of folds

# Perform RFE
set.seed(123)
rfe_result <- rfe(data[, -ncol(data)], data$y,
                  sizes = c(1:10),          # Subset sizes to evaluate
                  rfeControl = control)

# Display RFE results
cat("Selected Predictors:\n")
#> Selected Predictors:
print(predictors(rfe_result))
#> [1] "V1" "V2" "V3"

cat("\nModel Performance:\n")
#>
#> Model Performance:
print(rfe_result$results)
#>    Variables     RMSE  Rsquared      MAE    RMSESD RsquaredSD     MAESD
#> 1          1 2.342391 0.2118951 1.840089 0.5798450  0.2043123 0.4254393
#> 2          2 2.146397 0.3549917 1.726721 0.5579553  0.2434206 0.4490311
#> 3          3 2.063923 0.4209751 1.662022 0.5362445  0.2425727 0.4217594
#> 4          4 2.124035 0.3574168 1.698954 0.5469199  0.2261644 0.4212677
#> 5          5 2.128605 0.3526426 1.684169 0.5375931  0.2469891 0.4163273
#> 6          6 2.153916 0.3226917 1.712140 0.5036119  0.2236060 0.4058412
#> 7          7 2.162787 0.3223240 1.716382 0.5089243  0.2347677 0.3940646
#> 8          8 2.152186 0.3222999 1.698816 0.5064040  0.2419215 0.3826101
#> 9          9 2.137741 0.3288444 1.687136 0.5016526  0.2436893 0.3684675
#> 10        10 2.139102 0.3290236 1.684362 0.5127830  0.2457675 0.3715176

# Plot performance
plot(rfe_result, type = "l")
```

Interpretation

1. **Selected Predictors**:
   - The algorithm identifies the most relevant predictors based on their impact on model performance.
2. **Model Performance**:
   - The cross-validation results indicate how the model performs with different subset sizes, helping to select the optimal number of features.
3. **Feature Ranking**:
   - RFE ranks features by their importance, providing insights into which predictors are most influential.

Advantages:

- **Improved Interpretability**: Reduces the number of predictors, simplifying the model.
- **Performance Optimization**: Eliminates irrelevant or redundant features, improving predictive accuracy.
- **Flexibility**: Works with various model types (e.g., linear models, tree-based methods, SVMs).

Limitations:

- **Computationally Intensive**: Repeatedly trains models, which can be slow for large datasets or complex models.

- **Model Dependency**: Feature importance rankings depend on the underlying model, which may introduce bias.

Practical Considerations

- **When to use RFE?**
  - For high-dimensional datasets where feature selection is critical for model interpretability and performance.
  - In machine learning workflows where feature importance scores are available.
- **Extensions**:
  - Combine RFE with regularization methods (e.g., LASSO, Ridge) for additional feature selection.
  - Use advanced models (e.g., Random Forest, Gradient Boosting) for feature ranking.

15.3 Embedded Methods (Integrated into Model Training)

15.3.1 Regularization-Based Selection

- **Lasso (L1 Regularization)**: Shrinks some coefficients to zero, performing automatic selection.
- **Ridge (L2 Regularization)**: Shrinks coefficients but does not eliminate variables.
- **Elastic Net**: Combines L1 and L2 penalties for better feature selection.

15.3.2 Tree-Based Feature Importance

Decision trees and ensemble methods (Random Forests, Gradient Boosting) rank features based on their contribution to predictions.

15.3.3 Genetic Algorithms

Genetic Algorithms (GA) are inspired by the principles of natural selection and genetics. They are metaheuristic optimization techniques that iteratively evolve a population of solutions to find an optimal or near-optimal subset of predictors for regression or classification tasks.

Theoretical Foundation

1. **Objective**:

- Select a subset of predictors that optimizes a predefined fitness function (e.g., R^2, AIC, BIC, or prediction error).

2. **Key Concepts**:
   - **Population**: A collection of candidate solutions (subsets of predictors).
   - **Chromosome**: Represents a solution as a binary vector (e.g., `1` if a variable is included, `0` otherwise).
   - **Fitness Function**: Evaluates the quality of a solution based on the selected variables.
   - **Crossover**: Combines two solutions to create a new solution.
   - **Mutation**: Randomly alters a solution to maintain diversity in the population.
   - **Selection**: Chooses solutions with higher fitness to create the next generation.
3. **Advantages**:
   - Explores a wide solution space effectively.
   - Escapes local optima by introducing randomness.

Steps in Genetic Algorithms

1. **Initialization**:
   - Generate an initial population of candidate solutions randomly.
2. **Evaluation**:
   - Compute the fitness function for each solution in the population.
3. **Selection**:
   - Select solutions with higher fitness values to be parents for the next generation.
4. **Crossover**:
   - Combine pairs of parent solutions to create offspring.
5. **Mutation**:
   - Randomly modify some solutions to introduce variability.
6. **Replacement**:
   - Replace the old population with the new generation.
7. **Stopping Criteria**:
   - Terminate the algorithm after a fixed number of generations or when the improvement in fitness is below a threshold.

```r
# Install and load GA package
if (!requireNamespace("GA", quietly = TRUE))
    install.packages("GA")
library(GA)


# Simulated data
```

```r
set.seed(123)
n <- 100
p <- 10
X <- matrix(rnorm(n * p), nrow = n, ncol = p)

# Only first 3 variables are relevant
y <-
    5 + rowSums(X[, 1:3]) + rnorm(n, sd = 2)

# Convert to a data frame
data <- as.data.frame(X)
data$y <- y

# Define the fitness function
fitness_function <- function(binary_vector) {
    selected_vars <- which(binary_vector == 1)
    if (length(selected_vars) == 0)
        return(-Inf) # Penalize empty subsets
    model <- lm(y ~ ., data = data[, c(selected_vars, ncol(data))])
    - AIC(model)  # Return negative AIC (minimization problem)
}

# Run Genetic Algorithm
set.seed(123)
ga_result <-
    ga(
        type = "binary",
        # Binary encoding for variable selection
        fitness = fitness_function,
        nBits = p,
        # Number of predictors
        popSize = 50,
        # Population size
        maxiter = 100,
        # Maximum number of generations
        run = 10,
        # Stop if no improvement in 10 generations
        seed = 123
    )

# Extract the best solution
best_solution <- ga_result@solution[1, ]
selected_vars <- which(best_solution == 1)
cat("Selected Variables (Column Indices):\n", selected_vars, "\n")
#> Selected Variables (Column Indices):
```

```
#>  1 2 3 4

# Fit the final model with selected variables
final_model <-
    lm(y ~ ., data = data[, c(selected_vars, ncol(data))])
cat("Final Model Summary:\n")
#> Final Model Summary:
print(summary(final_model))
#>
#> Call:
#> lm(formula = y ~ ., data = data[, c(selected_vars, ncol(data))])
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -5.4013 -1.3823  0.0151  1.0796  5.1537
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   5.2726     0.2101  25.096  < 2e-16 ***
#> V1            1.1477     0.2290   5.012 2.49e-06 ***
#> V2            0.9469     0.2144   4.416 2.66e-05 ***
#> V3            0.6864     0.2199   3.121  0.00239 **
#> V4            0.3881     0.1997   1.943  0.05496 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 2.058 on 95 degrees of freedom
#> Multiple R-squared:  0.3574, Adjusted R-squared:  0.3304
#> F-statistic: 13.21 on 4 and 95 DF,  p-value: 1.352e-08
```

Interpretation

1. **Selected Variables**:
   - The genetic algorithm identifies the subset of predictors that minimizes the fitness function (negative AIC in this example).
2. **Model Performance**:
   - The final model can be evaluated using metrics such as adjusted R^2, prediction error, or cross-validation.
3. **Convergence**:
   - The algorithm evolves towards better solutions over generations, as indicated by improvements in the best fitness value.

Advantages:

- Can handle high-dimensional datasets and complex fitness functions.
- Avoids getting trapped in local optima by introducing randomness.
- Flexible: Can optimize any user-defined fitness function.

Limitations:

- Computationally intensive for large datasets or many predictors.
- Requires careful tuning of hyperparameters (e.g., population size, mutation rate).
- May not guarantee a globally optimal solution.

Practical Considerations

- **When to use Genetic Algorithms?**
  - For complex variable selection problems where traditional methods (e.g., stepwise selection) are insufficient.
  - When the fitness function is non-linear or involves interactions among predictors.
- **Tuning Tips**:
  - Adjust population size and mutation rate based on dataset size and complexity.
  - Use parallel computing to speed up the evaluation of fitness functions.

15.4 Summary Table

Method	Type	Approach	Key Criterion	Notes
Mallows's C Statistic	Information Criterion	Subset Selection	Model Complexity vs Fit	Balances fit and simplicity.
Akaike Information Criterion (AIC)	Information Criterion	Model Selection	Minimizes AIC	Penalizes model complexity.
Bayesian Information Criterion (BIC)	Information Criterion	Model Selection	Minimizes BIC	Stronger penalty for complexity.
Hannan-Quinn Criterion (HQC)	Information Criterion	Model Selection	Minimizes HQC	Combines AIC and BIC features.

Method	**Type**	**Approach**	**Key Criterion**	**Notes**
Minimum Description Length (MDL)	Information Criterion	Model Selection	Data + Model Encoding Costs	Focuses on encoding efficiency.
Prediction Error Sum of Squares (PRESS)	Error-Based	Cross-Validation	Minimizes Prediction Error	Measures predictive accuracy.
Best Subsets Algorithm	Exhaustive Search	Subset Selection	Best Fit Across Subsets	Considers all variable combinations.
Forward Selection	Stepwise	Add Variables	Significance Testing	Adds variables one at a time.
Backward Elimination	Stepwise	Remove Variables	Significance Testing	Removes variables iteratively.
Stepwise (Both Directions)	Stepwise	Add/Remove Variables	Significance Testing	Combines forward and backward methods.
Branch-and-Bound Algorithm	Optimized Search	Subset Selection	Efficient Subset Search	Avoids exhaustive search.
Recursive Feature Elimination (RFE)	Iterative Optimization	Feature Removal	Model Performance	Removes least important predictors.
Genetic Algorithms	Heuristic Search	Evolutionary Process	Fitness Function	Mimics natural selection for subsets.

Chapter 16

Hypothesis Testing

Hypothesis testing is one of the cornerstones of statistical inference, used widely across disciplines such as economics, finance, psychology, and more. Researchers employ hypothesis testing to draw conclusions about population parameters based on sample data. Central to this process is the concept of the *p-value*, which helps quantify how unlikely the observed data (or more extreme data) would be if the null hypothesis were true.

However, as data collection has become easier and cheaper—especially in the age of big data—there is a growing awareness that *large sample sizes (large n)* can inflate the likelihood of finding statistically significant, but practically negligible, effects. Moreover, this can lead to "p-value hacking," where researchers run numerous tests or adopt flexible analytical approaches until they find a (sometimes minuscule) effect that achieves a conventional significance level (often $p < .05$).

16.1 Null Hypothesis Significance Testing

Null Hypothesis Significance Testing (NHST) is the foundation of statistical inference. It provides a structured approach to evaluating whether observed data provides sufficient evidence to reject a null hypothesis (H_0) in favor of an alternative hypothesis (H_a).

NHST follows these key steps:

1. **Define Hypotheses**
   - The null hypothesis (H_0) represents the default assumption (e.g., no effect, no difference).
   - The alternative hypothesis (H_a) represents the competing claim (e.g., a nonzero effect, a relationship between variables).
2. **Select a Test Statistic**
   - The test statistic (e.g., T, W, F) quantifies evidence against H_0.

- It follows a known distribution under H_0 (e.g., normal, chi-square, F-distribution).

3. **Decision Rule & p-value**
   - If the test statistic **exceeds a critical value** or the **p-value is below** α, we reject H_0.
   - Otherwise, we fail to reject H_0, meaning the evidence is insufficient to rule it out.

16.1.1 Error Types in Hypothesis Testing

In hypothesis testing, we may **incorrectly** reject or fail to reject the null hypothesis, leading to two types of errors:

- **Type I Error (False Positive)**:
  - **Rejecting** H_0 when it is actually true.
  - Example: Concluding an effect exists when it does not.
- **Type II Error (False Negative)**:
  - **Failing to reject** H_0 when it is actually false.
  - Example: Missing a real effect because the test lacked power.

The **power of a test** is the probability of correctly rejecting H_0 when it is false:

$$\text{Power} = 1 - P(\text{Type II Error})$$

A **higher power** (typically ≥ 0.8) reduces Type II errors and increases the likelihood of detecting true effects.

16.1.2 Hypothesis Testing Framework

Hypothesis tests can be **two-sided** or **one-sided**, depending on the research question.

16.1.2.1 Two-Sided Test

In a two-sided test, we examine whether a parameter is significantly different from a hypothesized value (usually zero):

$$H_0 : \beta_j = 0$$
$$H_1 : \beta_j \neq 0$$

Under the null hypothesis, and assuming standard ordinary least squares assumptions (A1-A3a, A5), the asymptotic distribution of the OLS estimator is:

$$\sqrt{n}\hat{\beta}_j \sim N(0, \text{Avar}(\sqrt{n}\hat{\beta}_j))$$

where $\text{Avar}(\cdot)$ denotes the asymptotic variance.

16.1.2.2 One-Sided Test

For a one-sided hypothesis test, the null hypothesis includes a range of values, and we test against a directional alternative:

$$H_0 : \beta_j \geq 0$$
$$H_1 : \beta_j < 0$$

The "hardest" null value to reject is $\beta_j = 0$. Under this specific null, the estimator follows the same asymptotic distribution:

$$\sqrt{n}\hat{\beta}_j \sim N(0, \text{Avar}(\sqrt{n}\hat{\beta}_j))$$

16.1.3 Interpreting Hypothesis Testing Results

When conducting hypothesis tests, it is essential to distinguish between **population parameters** and **sample estimates**:

- Hypotheses are always written in terms of the population parameter (β), not the sample estimate ($\hat{\beta}$).
- Some disciplines use different notations:
  - β: Standardized coefficient (useful for comparing relative effects, scale-free).
  - $\mathbf{b}$: Unstandardized coefficient (more interpretable in practical applications, e.g., policy decisions).

The relationship between these coefficients is:

$$\beta_j = \mathbf{b}_j \frac{s_{x_j}}{s_y}$$

where s_{x_j} and s_y are the standard deviations of the independent and dependent variables.

16.1.4 Understanding p-Values

The p-value is the probability, under the assumption that H_0 is true, of observing a test statistic at least as extreme as the one computed from the sample data. Formally,

$$p\text{-value} = P(\text{Test Statistic} \geq \text{observed value} \mid H_0 \text{ is true})$$

Interpretation

- A small p-value indicates that *if* H_0 were true, seeing the observed data (or something more extreme) would be unlikely.
- By convention, if $p < \alpha$ (often 0.05), the result is deemed "statistically significant," and we reject H_0.
- **Important Caveat**: "Statistically significant" is *not* the same as "practically significant" or "economically significant." A difference can be statistically significant yet trivial in magnitude, with negligible real-world implications.

Misconceptions

- The p-value is *not* the probability that H_0 is true or false.
- A p-value above 0.05 does *not* prove that there is "no effect." It simply suggests that the data do not provide sufficient evidence (at the chosen significance level) to reject H_0.
- A p-value below 0.05 does *not* prove that an effect is "real" or large. It indicates that the data are unusual enough under H_0 that we decide to reject H_0, given our chosen threshold.

16.1.5 The Role of Sample Size

A critical factor influencing the outcome of hypothesis tests is *sample size* (n).

Increasing Power with Large n

- **Statistical Power**: The probability of correctly rejecting H_0 when H_0 is false. Large sample sizes increase statistical power, making it easier to detect even tiny deviations from H_0.
- **Implication**: If the true effect size in the population is very small (e.g., a 0.2% difference in average returns between two trading strategies), a study with a large enough n might still find it statistically significant (p-value $<$ 0.05).

Tendency Toward Over-Sensitivity

As n grows, the standard errors decrease. Thus, even minuscule differences from the null hypothesis become less likely to be attributed to random chance, yielding low p-values. This can lead to findings that are *statistically* significant but have *negligible* real-world impact.

- **Example**: Suppose an economist is testing if a policy intervention changes employment rates by 0.1%. With a small sample size, the test might not detect this difference. But with a massive dataset, the same 0.1% difference might yield a p-value < 0.05, even though a 0.1% change may not be economically meaningful.

16.1.6 p-Value Hacking

p-Hacking refers to the process of manipulating data analysis until a statistically significant result (p-value < 0.05) is achieved. This can include:

- Running multiple tests on the same dataset and only reporting those that yield significance.
- Stopping data collection once a significant p-value is reached.
- Trying various model specifications (e.g., adding or removing control variables) until one finds a significant effect.
- Selectively reporting outcomes (publication bias).

With large datasets, the "search space" for potential analyses grows exponentially. If researchers test many hypotheses or sift through a wide range of variables and subgroups, they can almost always find a "significant" result by chance alone.

- **Multiple Comparison Problem**: When multiple tests are conducted, the chance of finding at least one "significant" result purely by coincidence increases. For instance, with 20 independent tests at $\alpha = .05$, there is a 64% chance ($1-0.95^{20}$) of incorrectly rejecting at least one null hypothesis.

16.1.7 Practical vs. Statistical Significance

In economics and finance, it is crucial to distinguish between results that are *statistically significant* and those that are *economically meaningful*. Economic or financial significance asks: *Does this effect have tangible importance to policymakers, businesses, or investors?*

- A result might show that a new trading algorithm yields returns that are statistically different from zero, but if that difference is 0.0001% on average, it might not be profitable after accounting for transaction fees, taxes, or other frictions—hence lacking *economic significance*.

16.1.8 Mitigating the Misuse of p-Values

16.1.8.1 Pre-Registration and Replication

- **Pre-Registration**: Researchers specify hypotheses and analytical methods before seeing the data, reducing the temptation to p-hack.
- **Replication**: Independent replication studies help confirm whether a result is robust or merely a fluke.

16.1.8.2 Using Alternatives to (or Supplements for) p-Values

- **Bayesian Methods**: Provide posterior probabilities that incorporate prior information, often giving a more nuanced understanding of uncertainty.
- **Effect Size & Confidence Intervals**: Shift the focus from "Is it significant?" to "How large is the effect, and what is its plausible range?"
- Equivalence Testing: Sometimes the goal is to show the effect is *not* larger than a certain threshold. Equivalence tests can be used to conclude "no clinically (or economically) significant difference."

16.1.8.3 Adjusting for Multiple Comparisons

- **Bonferroni Correction**: Requires using a more stringent significance threshold when multiple tests are performed (e.g., α/m for m tests).
- False Discovery Rate Control: Allows a more flexible approach, controlling the expected proportion of false positives among significant findings.

16.1.8.4 Emphasizing Relevance Over Statistical "Stars"

Encourage journals, reviewers, and academic circles to stress the *magnitude of effects* and *robustness checks* over whether the result crosses a conventional p-value threshold (like 0.05).

There are three commonly used methods for hypothesis testing:

1. Likelihood Ratio Test: Compares the likelihood under the null and alternative models. Often used for nested models.
2. Wald Test: Assesses whether an estimated parameter is significantly different from a hypothesized value. Requires only one maximization (under the full model).

3. Lagrange Multiplier (Score) Test: Evaluates the slope of the likelihood function at the null hypothesis value. Performs well in small to moderate samples.

16.1.9 Wald Test

The Wald test assesses whether estimated parameters are significantly different from hypothesized values, based on the asymptotic distribution of the estimator.

The general form of the Wald statistic is:

$$W = (\hat{\theta} - \theta_0)'[cov(\hat{\theta})]^{-1}(\hat{\theta} - \theta_0)$$
$$W \sim \chi^2_q$$

where:

- $cov(\hat{\theta})$ is given by the inverse Fisher Information matrix evaluated at $\hat{\theta}$,
- q is the rank of $cov(\hat{\theta})$, which corresponds to the number of non-redundant parameters in θ.

The Wald statistic can also be expressed in different ways:

1. Quadratic form of the test statistic:

$$t_W = \frac{(\hat{\theta} - \theta_0)^2}{I(\theta_0)^{-1}} \sim \chi^2_{(v)}$$

where v is the degree of freedom.

2. Standardized Wald test statistic:

$$s_W = \frac{\hat{\theta} - \theta_0}{\sqrt{I(\hat{\theta})^{-1}}} \sim Z$$

This represents how far the sample estimate is from the hypothesized population parameter.

Significance Level and Confidence Level

- The significance level (α) is the probability threshold at which we reject the null hypothesis.
- The confidence level ($1 - \alpha$) determines the range within which the population parameter is expected to fall with a given probability.

To standardize the estimator and null value, we define the test statistic for the OLS estimator:

$$T = \frac{\sqrt{n}(\hat{\beta}_j - \beta_{j0})}{\sqrt{n}SE(\hat{\beta}_j)} \sim^a N(0,1)$$

Equivalently:

$$T = \frac{(\hat{\beta}_j - \beta_{j0})}{SE(\hat{\beta}_j)} \sim^a N(0,1)$$

where:

- T is the test statistic (a function of the data and null hypothesis),
- t is the observed realization of T.

16.1.9.1 Evaluating the Test Statistic

There are three equivalent methods for evaluating hypothesis tests:

1. Critical Value Method

For a given significance level α, determine the critical value (c):

- One-sided test: $H_0 : \beta_j \geq \beta_{j0}$

$$P(T < c|H_0) = \alpha$$

Reject H_0 if $t < c$.

- One-sided test: $H_0 : \beta_j \leq \beta_{j0}$

$$P(T > c|H_0) = \alpha$$

Reject H_0 if $t > c$.

- Two-sided test: $H_0 : \beta_j \neq \beta_{j0}$

$$P(|T| > c|H_0) = \alpha$$

Reject H_0 if $|t| > c$.

2. p-value Method

The p-value is the probability of observing a test statistic as extreme as the one obtained, given that the null hypothesis is true.

- One-sided test: $H_0 : \beta_j \geq \beta_{j0}$

$$\text{p-value} = P(T < t|H_0)$$

- One-sided test: $H_0 : \beta_j \leq \beta_{j0}$

$$\text{p-value} = P(T > t|H_0)$$

- Two-sided test: $H_0 : \beta_j \neq \beta_{j0}$

$$\text{p-value} = P(|T| > |t||H_0)$$

Reject H_0 if p-value $< \alpha$.

3. Confidence Interval Method

Using the critical value associated with a given significance level, construct a confidence interval:

$$CI(\hat{\beta}_j)_\alpha = \left[\hat{\beta}_j - c \times SE(\hat{\beta}_j), \hat{\beta}_j + c \times SE(\hat{\beta}_j)\right]$$

Reject H_0 if the hypothesized value falls outside the confidence interval.

- We are not testing whether the true population value is close to the estimate. Instead, we are testing:
  Given a fixed true population value of the parameter, how likely is it that we observed this estimate?

- This can be interpreted as:
  We believe with $(1 - \alpha) \times 100\%$ probability that the confidence interval captures the true parameter value.

Finite Sample Properties

Under stronger assumptions (A1-A6), we can consider finite sample properties:

$$T = \frac{\hat{\beta}_j - \beta_{j0}}{SE(\hat{\beta}_j)} \sim T(n - k)$$

where:

- The derivation of this distribution depends strongly on:

  - A4 (Homoskedasticity)
  - A5 (Data Generation via Random Sampling)
- The T-statistic follows a Student's t-distribution because:
  - The numerator is normally distributed.
- The denominator follows a χ^2 distribution.
- Critical values and p-values will be computed using the Student's t-distribution instead of the standard normal distribution.
- As $n \to \infty$, the $T(n-k)$ distribution converges to a standard normal distribution.

Rule of Thumb

- If $n-k > 120$:
  - The t-distribution critical values and p-values closely approximate those from the standard normal distribution.
- If $n-k < 120$:
  - If (A1-A6) hold, the t-test is an exact finite-sample test.
  - If (A1-A3a, A5) hold, the t-distribution is asymptotically normal.
    * Using the t-distribution for critical values is a valid asymptotic test.
    * The discrepancy in critical values disappears as $n \to \infty$.

16.1.9.2 Multiple Hypothesis Testing

We often need to test multiple parameters simultaneously:

- Example 1: $H_0 : \beta_1 = 0$ and $\beta_2 = 0$
- Example 2: $H_0 : \beta_1 = 1$ and $\beta_2 = 0$

Performing separate hypothesis tests on individual parameters does not answer the question of joint significance.
We need a test that accounts for joint distributions rather than evaluating two marginal distributions separately.

Consider the multiple regression model:

$$y = \beta_0 + x_1\beta_1 + x_2\beta_2 + x_3\beta_3 + \epsilon$$

The null hypothesis $H_0 : \beta_1 = 0$ and $\beta_2 = 0$ can be rewritten in matrix form as:

$$H_0 : \mathbf{R}\beta - \mathbf{q} = 0$$

where:

- $\mathbf{R}$ is an $m \times k$ matrix, where:
  - m = number of restrictions.
  - k = number of parameters.
- $\mathbf{q}$ is a $k \times 1$ vector that contains the null hypothesis values.

For the example $H_0 : \beta_1 = 0$ and $\beta_2 = 0$, we define:

$$\mathbf{R} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

For the OLS estimator under multiple hypotheses, we use the F-statistic:

$$F = \frac{(\mathbf{R}\hat{\boldsymbol{\beta}} - \mathbf{q})'\hat{\Sigma}^{-1}(\mathbf{R}\hat{\boldsymbol{\beta}} - \mathbf{q})}{m} \sim^a F(m, n-k)$$

where:

- $\hat{\Sigma}^{-1}$ is the estimator for the asymptotic variance-covariance matrix.
- m is the number of restrictions.
- $n - k$ is the residual degrees of freedom.

Assumptions for Variance Estimation

- If A4 (Homoskedasticity) holds:
  - Both the homoskedastic and heteroskedastic variance estimators are valid.
- If A4 does not hold:
  - Only the heteroskedastic variance estimator remains valid.

Relationship Between F and t-Tests

- When $m = 1$ (only one restriction), the F-statistic is simply the squared t-statistic:

$$F = t^2$$

- Since the F-distribution is strictly positive, it is one-sided by definition.

16.1.9.3 Linear Combination Testing

When testing multiple parameters simultaneously, we often assess linear combinations of parameters rather than testing them individually.

For example, consider the following hypotheses:

$$H_0 : \beta_1 - \beta_2 = 0$$
$$H_0 : \beta_1 - \beta_2 > 0$$
$$H_0 : \beta_1 - 2\beta_2 = 0$$

Each of these represents a single restriction on a function of the parameters.

The null hypothesis:

$$H_0 : \beta_1 - \beta_2 = 0$$

can be rewritten in matrix form as:

$$H_0 : \mathbf{R}\beta - \mathbf{q} = 0$$

where:

$$\mathbf{R} = \begin{bmatrix} 0 & 1 & -1 & 0 & 0 \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} 0 \end{bmatrix}$$

Interpretation:

- $\mathbf{R}$ is a $1 \times k$ matrix that selects the relevant parameters for the hypothesis.
- $\mathbf{q}$ is a $k \times 1$ vector containing the hypothesized values of the linear combination.
- This formulation allows us to use a generalized Wald test to assess whether the constraint holds.

The Wald test statistic for a linear hypothesis:

$$W = \frac{(\mathbf{R}\hat{\beta} - \mathbf{q})' \left(\mathbf{R}\hat{\Sigma}\mathbf{R}'\right)^{-1} (\mathbf{R}\hat{\beta} - \mathbf{q})}{s^2 q} \sim F_{q,n-k}$$

where:

- $\hat{\beta}$ is the vector of estimated coefficients.
- $\hat{\Sigma}$ is the variance-covariance matrix of $\hat{\beta}$.
- s^2 is the estimated error variance.
- q is the number of restrictions.
- The test follows an F-distribution with degrees of freedom $(q, n-k)$.

```r
library(car)

# Fit a multiple regression model
mod.duncan <- lm(prestige ~ income + education, data=Duncan)

# Test whether income and education coefficients are equal
linearHypothesis(mod.duncan, "1*income - 1*education = 0")
#> Linear hypothesis test
#>
#> Hypothesis:
#> income - education = 0
#>
#> Model 1: restricted model
#> Model 2: prestige ~ income + education
#>
#>   Res.Df    RSS Df Sum of Sq      F Pr(>F)
#> 1     43 7518.9
#> 2     42 7506.7  1    12.195 0.0682 0.7952
```

- This tests whether $\beta_1 = \beta_2$ (i.e., whether income and education have the same effect on prestige).
- If the p-value is low, we reject the null hypothesis and conclude that income and education contribute differently to prestige.

16.1.9.4 Estimating the Difference Between Two Coefficients

In some cases, we may be interested in comparing two regression coefficients directly rather than evaluating them separately. For example, we might want to test:

$$H_0 : \beta_1 = \beta_2$$

which is equivalent to testing whether their difference is zero:

$$H_0 : \beta_1 - \beta_2 = 0$$

Alternatively, we can directly estimate the difference between two regression coefficients.

```r
difftest_lm <- function(x1, x2, model) {
    # Compute coefficient difference
    diffest <-
        summary(model)$coef[x1, "Estimate"] - summary(model)$coef[x2, "Estimate"]

    # Compute variance of the difference
```

```
    vardiff <- (summary(model)$coef[x1, "Std. Error"] ^ 2 +
                    summary(model)$coef[x2, "Std. Error"] ^ 2) - (2 * vcov(model)[x1, x

    # Compute standard error of the difference
    diffse <- sqrt(vardiff)

    # Compute t-statistic
    tdiff <- diffest / diffse

    # Compute p-value (two-sided test)
    ptdiff <- 2 * (1 - pt(abs(tdiff), model$df.residual))

    # Compute confidence interval
    upr <- diffest + qt(0.975, df = model$df.residual) * diffse
    lwr <- diffest - qt(0.975, df = model$df.residual) * diffse

    # Return results as a named list
    return(
        list(
            estimate = round(diffest, 2),
            t_stat = round(tdiff, 2),
            p_value = round(ptdiff, 4),
            lower_CI = round(lwr, 2),
            upper_CI = round(upr, 2),
            df = model$df.residual
        )
    )
}
```

We demonstrate this function using the Duncan dataset from the {car} package:

```
library(car)

# Load Duncan dataset
data(Duncan)

# Fit a linear regression model
mod.duncan <- lm(prestige ~ income + education, data = Duncan)

# Compare the effects of income and education
difftest_lm("income", "education", mod.duncan)
#> $estimate
#> [1] 0.05
#>
#> $t_stat
#> [1] 0.26
```

```
#>
#> $p_value
#> [1] 0.7952
#>
#> $lower_CI
#> [1] -0.36
#>
#> $upper_CI
#> [1] 0.46
#>
#> $df
#> [1] 42
```

16.1.9.5 Nonlinear Hypothesis Testing

In many applications, we may need to test nonlinear restrictions on parameters. These can be expressed as a set of q nonlinear functions:

$$\mathbf{h}(\theta) = \{h_1(\theta), ..., h_q(\theta)\}'$$

where each $h_j(\theta)$ is a nonlinear function of the parameter vector θ.

To approximate nonlinear restrictions, we use the Jacobian matrix, denoted as $\mathbf{H}(\theta)$, which contains the first-order partial derivatives of $\mathbf{h}(\theta)$ with respect to the parameters:

$$\mathbf{H}_{q\times p}(\theta) = \begin{bmatrix} \frac{\partial h_1(\theta)}{\partial \theta_1} & \cdots & \frac{\partial h_1(\theta)}{\partial \theta_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_q(\theta)}{\partial \theta_1} & \cdots & \frac{\partial h_q(\theta)}{\partial \theta_p} \end{bmatrix}$$

where:

- q is the number of nonlinear restrictions,
- p is the number of estimated parameters.

The Jacobian matrix linearizes the nonlinear restrictions and allows for an approximation of the hypothesis test using a Wald statistic.

We test the null hypothesis:

$$H_0 : \mathbf{h}(\theta) = 0$$

against the two-sided alternative using the Wald statistic:

$$W = \frac{\mathbf{h}(\hat{\ })' \left\{ \mathbf{H}(\hat{\theta}) \left[\mathbf{F}(\hat{\theta})' \mathbf{F}(\hat{\theta}) \right]^{-1} \mathbf{H}(\hat{\theta})' \right\}^{-1} \mathbf{h}(\hat{\theta})}{s^2 q} \sim F_{q,n-p}$$

where:

- $\hat{\theta}$ is the estimated parameter vector,
- $\mathbf{H}(\hat{\theta})$ is the Jacobian matrix evaluated at $\hat{\theta}$,
- $\mathbf{F}(\hat{\theta})$ is the Fisher Information Matrix,
- s^2 is the estimated error variance,
- q is the number of restrictions,
- n is the sample size,
- p is the number of parameters.

The test statistic follows an F-distribution with degrees of freedom $(q, n-p)$.

```r
library(car)
library(nlWaldTest)

# Load example data
data(Duncan)

# Fit a multiple regression model
mod.duncan <- lm(prestige ~ income + education, data = Duncan)

# Define a nonlinear hypothesis: income squared equals education
nl_hypothesis <- "b[2]^2 - b[3] = 0"

# Conduct the nonlinear Wald test
nlWaldtest(mod.duncan, texts = nl_hypothesis)
#>
#>  Wald Chi-square test of a restriction on model parameters
#>
#> data:  mod.duncan
#> Chisq = 0.69385, df = 1, p-value = 0.4049
```

- If the Wald statistic is large, we reject H_0 and conclude that the nonlinear restriction does not hold.
- The p-value provides the probability of observing such an extreme test statistic under the null hypothesis.

- The F-distribution accounts for the fact that multiple nonlinear restrictions are being tested.

16.1.10 Likelihood Ratio Test

The Likelihood Ratio Test (LRT) is a general method for comparing two nested models:

- The reduced model under the null hypothesis (H_0), which imposes constraints on parameters.
- The full model, which allows more flexibility under the alternative hypothesis (H_a).

The test evaluates how much more likely the data is under the full model compared to the restricted model.

The likelihood ratio test statistic is given by:

$$t_{LR} = 2[l(\hat{\theta}) - l(\theta_0)] \sim \chi^2_v$$

where:

- $l(\hat{\theta})$ is the log-likelihood evaluated at the estimated parameter $\hat{\theta}$ (from the full model),
- $l(\theta_0)$ is the log-likelihood evaluated at the hypothesized parameter θ_0 (from the reduced model),
- v is the degrees of freedom (the difference in the number of parameters between the full and reduced models).

This test compares the height of the log-likelihood of the sample estimate versus the hypothesized population parameter.

This test also considers the ratio of two maximized likelihoods:

$$\begin{aligned} L_r &= \text{maximized likelihood under } H_0 \text{ (reduced model)} \\ L_f &= \text{maximized likelihood under } H_0 \cup H_a \text{ (full model)} \end{aligned}$$

Then, the likelihood ratio is defined as:

$$\Lambda = \frac{L_r}{L_f}$$

where:

- Λ cannot exceed 1, because L_f (the likelihood of the full model) is always at least as large as L_r.

The likelihood ratio test statistic is then:

$$-2\ln(\Lambda) = -2\ln\left(\frac{L_r}{L_f}\right) = -2(l_r - l_f)$$
$$\lim_{n\to\infty}(-2\ln(\Lambda)) \sim \chi^2_v$$

where:

- v is the difference in the number of parameters between the full and reduced models.

If the likelihood ratio is small (i.e., L_r is much smaller than L_f), then:

- The test statistic exceeds the critical value from the χ^2_v distribution.
- We reject the reduced model and accept the full model at the $\alpha \times 100\%$ significance level.

```r
library(lmtest)

# Load example dataset
data(mtcars)

# Fit a full model with two predictors
full_model <- lm(mpg ~ hp + wt, data = mtcars)

# Fit a reduced model with only one predictor
reduced_model <- lm(mpg ~ hp, data = mtcars)

# Perform the likelihood ratio test
lrtest(reduced_model, full_model)
#> Likelihood ratio test
#>
#> Model 1: mpg ~ hp
#> Model 2: mpg ~ hp + wt
#>   #Df  LogLik Df  Chisq Pr(>Chisq)
#> 1   3 -87.619
#> 2   4 -74.326  1 26.586   2.52e-07 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- If the p-value is small, the reduced model is significantly worse, and we reject H_0.

- A large test statistic indicates that removing a predictor leads to a substantial drop in model fit.

16.1.11 Lagrange Multiplier (Score) Test

The Lagrange Multiplier (LM) Test, also known as the Score Test, evaluates whether a restricted model (under H_0) significantly underperforms compared to an unrestricted model (under H_a) without estimating the full model.

Unlike the Likelihood Ratio Test, which requires estimating both models, the LM test only requires estimation under the restricted model (H_0).

The LM test statistic is based on the first derivative (score function) of the log-likelihood function, evaluated at the parameter estimate under the null hypothesis (θ_0):

$$t_S = \frac{S(\theta_0)^2}{I(\theta_0)} \sim \chi^2_v$$

where:

- $S(\theta_0) = \left.\frac{\partial l(\theta)}{\partial \theta}\right|_{\theta=\theta_0}$ is the score function, i.e., the first derivative of the log-likelihood function evaluated at θ_0.
- $I(\theta_0)$ is the Fisher Information Matrix, which quantifies the curvature (second derivative) of the log-likelihood.
- v is the degrees of freedom, equal to the number of constraints imposed by H_0.

This test compares:

- The slope of the log-likelihood function at θ_0 (which should be flat under H_0).
- The curvature of the log-likelihood function (captured by $I(\theta_0)$).

Interpretation of the LM Test

- If t_S is large, the slope of the log-likelihood function at θ_0 is steep, indicating that the model fit improves significantly when moving away from θ_0.
- If t_S is small, the log-likelihood function remains nearly flat at θ_0, meaning that the additional parameters in the unrestricted model do not substantially improve the fit.

If the score function $S(\theta_0)$ is significantly different from zero, then we reject H_0 because it suggests that the likelihood function is increasing, implying a better model fit when moving away from θ_0.

```r
# Load necessary libraries
library(lmtest)  # For the Lagrange Multiplier test
library(car)     # For example data

# Load example data
data(Prestige)

# Fit a linear regression model
model <- lm(prestige ~ income + education, data = Prestige)

# Perform the Lagrange Multiplier test for heteroscedasticity
# Using the Breusch-Pagan test (a type of LM test)
lm_test <- bptest(model)

# Print the results
print(lm_test)
#>
#>  studentized Breusch-Pagan test
#>
#> data:  model
#> BP = 4.1838, df = 2, p-value = 0.1235
```

- `bptest`: This function from the `lmtest` package performs the Breusch-Pagan test, which is a Lagrange Multiplier test for heteroscedasticity.
- Null Hypothesis: The null hypothesis is that the variance of the residuals is constant (homoscedasticity).
- Alternative Hypothesis: The alternative hypothesis is that the variance of the residuals is not constant (heteroscedasticity).

16.1.12 Comparing Hypothesis Tests

A visual comparison of hypothesis tests is shown below:

```r
# Load required libraries
library(ggplot2)

# Generate data for a normal likelihood function
theta <- seq(-3, 3, length.out = 200)  # Theta values

# Likelihood function with theta_hat = 1
likelihood <-
    dnorm(theta, mean = 1, sd = 1)
```

```r
df <- data.frame(theta, likelihood)

# Define key points
theta_0   <- 0  # Null hypothesis value
theta_hat <- 1  # Estimated parameter (full model)
likelihood_0 <-
    dnorm(theta_0, mean = 1, sd = 1)  # Likelihood at theta_0
likelihood_hat <-
    dnorm(theta_hat, mean = 1, sd = 1)  # Likelihood at theta_hat

# Plot likelihood function
ggplot(df, aes(x = theta, y = likelihood)) +
    geom_line(color = "blue", linewidth = 1.2) +  # Likelihood curve

    # Vertical lines for theta_0 and theta_hat
    geom_vline(
        xintercept = theta_0,
        linetype = "dashed",
        color = "black",
        linewidth = 1
    ) +
    geom_vline(
        xintercept = theta_hat,
        linetype = "dashed",
        color = "red",
        linewidth = 1
    ) +

    # Labels for theta_0 and theta_hat
    annotate(
        "text",
        x = theta_0 - 0.1,
        y = -0.02,
        label = expression(theta[0]),
        color = "black",
        size = 5,
        fontface = "bold"
    ) +
    annotate(
        "text",
        x = theta_hat + 0.1,
        y = -0.02,
        label = expression(hat(theta)),
        color = "red",
        size = 5,
```

```
        fontface = "bold"
    ) +

    # LRT: Compare heights of likelihood at theta_0 and theta_hat
    annotate(
        "segment",
        x = theta_0,
        xend = theta_0,
        y = likelihood_0,
        yend = likelihood_hat,
        color = "purple",
        linewidth = 1.2,
        arrow = arrow(length = unit(0.15, "inches"))
    ) +
    annotate(
        "text",
        x = -2,
        y = (likelihood_0 + likelihood_hat) / 2 + 0.02,
        label = "LRT: Height",
        color = "purple",
        hjust = 0,
        fontface = "bold",
        size = 5
    ) +
    # Add horizontal lines at both ends of LRT height comparison
    annotate(
        "segment",
        x = -2.5,
        xend = 2.5,
        y = likelihood_0,
        yend = likelihood_0,
        color = "purple",
        linetype = "dotted",
        linewidth = 1
    ) +
    annotate(
        "segment",
        x = -2.5,
        xend = 2.5,
        y = likelihood_hat,
        yend = likelihood_hat,
        color = "purple",
        linetype = "dotted",
        linewidth = 1
    ) +
```

```r
# Wald Test: Distance between theta_0 and theta_hat
annotate(
    "segment",
    x = theta_0,
    xend = theta_hat,
    y = 0.05,
    yend = 0.05,
    color = "green",
    linewidth = 1.2,
    arrow = arrow(length = unit(0.15, "inches"))
) +
annotate(
    "text",
    x = (theta_0 + theta_hat) / 2,
    y = 0.07,
    label = "Wald: Distance",
    color = "green",
    hjust = 0.5,
    fontface = "bold",
    size = 5
) +

# LM Test: Slope at theta_0
annotate(
    "segment",
    x = theta_0 - 0.2,
    xend = theta_0 + 0.2,
    y = dnorm(theta_0 - 0.2, mean = 1, sd = 1),
    yend = dnorm(theta_0 + 0.2, mean = 1, sd = 1),
    color = "orange",
    linewidth = 1.2,
    arrow = arrow(length = unit(0.15, "inches"))
) +
annotate(
    "text",
    x = -1.5,
    y = dnorm(-1, mean = 1, sd = 1) + .2,
    label = "LM: Slope",
    color = "orange",
    hjust = 0,
    fontface = "bold",
    size = 5
) +

# Titles and themes
```

```
    theme_minimal() +
    labs(title = "Comparison of Hypothesis Tests",
         x = expression(theta),
         y = "Likelihood") +
    theme(
        plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
        axis.title = element_text(size = 14),
        axis.text = element_text(size = 12)
    )
```

Figure adapted from (Fox, 1997).

Each test approaches hypothesis evaluation differently:

1. Likelihood Ratio Test: Compares the **heights** of the log-likelihood at $\hat{\theta}$ (full model) vs. θ_0 (restricted model).
2. Wald Test: Measures the **distance** between $\hat{\theta}$ and θ_0.
3. Lagrange Multiplier Test: Examines the **slope** of the log-likelihood at θ_0 to check if movement towards $\hat{\theta}$ significantly improves fit.

The Likelihood Ratio Test and Lagrange Multiplier Test perform well in small to moderate samples, while the Wald Test is computationally simpler as it only requires one model estimation.

Test	Key Idea	Computation	Best Use Case
Likelihood Ratio Test	Compares log-likelihoods of full vs. restricted models	Estimates both models	When both models can be estimated
Wald Test	Checks if parameters significantly differ from H_0	Estimates only the full model	When the full model is available

Test	Key Idea	Computation	Best Use Case
Lagrange Multiplier Test	Tests if the score function suggests moving away from H_0	Estimates only the restricted model	When the full model is difficult to estimate

---

16.2 Two One-Sided Tests Equivalence Testing

The Two One-Sided Tests (TOST) procedure is a method used in equivalence testing to determine whether a population effect size falls within a range of practical equivalence.

Unlike traditional null hypothesis significance testing (NHST), which focuses on detecting differences, TOST tests for similarity by checking whether an effect is small enough to be practically insignificant.

16.2.1 When to Use TOST?

1. Bioequivalence Testing
   - Example: Determining whether a generic drug is equivalent to a brand-name drug in terms of effectiveness.
2. Non-Inferiority Testing
   - Example: Assessing whether a new teaching method is not worse than a traditional method by a meaningful margin.
3. Equivalence in Business & Finance
   - Example: Comparing the performance of two financial models to determine if they produce practically the same results.
4. Psychological & Behavioral Research
   - Example: Determining whether a new intervention is equally effective as an existing one.

In traditional hypothesis testing, we assess:

$$H_0 : \theta = \theta_0 \quad vs. \quad H_a : \theta \neq \theta_0$$

where θ is a population parameter (e.g., mean difference, regression coefficient, or effect size).

However, in equivalence testing, we are interested in whether θ falls within a predefined equivalence margin $(-\Delta, \Delta)$.

This leads to the TOST procedure, where we conduct two one-sided tests:

1st One-Sided Test:

$$H_0 : \theta \leq -\Delta \quad vs. \quad H_a : \theta > -\Delta$$

2nd One-Sided Test:

$$H_0 : \theta \geq \Delta \quad vs. \quad H_a : \theta < \Delta$$

If both null hypotheses are rejected, then we conclude equivalence (i.e., θ is within the equivalence range).

16.2.2 Interpretation of the TOST Procedure

- If the p-value for both one-sided tests is less than α, then we conclude that the effect size falls within the equivalence bounds.
- If one or both p-values are greater than α, we fail to reject the null hypothesis and cannot claim equivalence.
- The TOST procedure provides stronger evidence of similarity than traditional NHST, which only assesses whether an effect is *statistically different from zero* rather than *practically insignificant*.

16.2.3 Relationship to Confidence Intervals

Another way to interpret TOST is through confidence intervals (CIs):

- If the entire $(1 - 2\alpha) \times 100\%$ confidence interval lies within $[-\Delta, \Delta]$, we conclude equivalence.
- If the confidence interval extends beyond the equivalence range, we fail to establish equivalence.

This relationship ensures that TOST is consistent with CI-based inference.

16.2.4 Example 1: Testing the Equivalence of Two Means

Suppose we have two groups and want to test whether their mean difference is practically insignificant within a range of $[-0.5, 0.5]$.

```
library(TOSTER)

# Simulated data: Two groups with similar means
set.seed(123)
group1 <- rnorm(30, mean = 5, sd = 1)
```

```r
group2 <- rnorm(30, mean = 5.1, sd = 1)

# Perform TOST equivalence test
TOSTtwo(
    m1 = mean(group1),
    sd1 = sd(group1),
    n1 = length(group1),
    m2 = mean(group2),
    sd2 = sd(group2),
    n2 = length(group2),
    low_eqbound = -0.5,
    high_eqbound = 0.5,
    alpha = 0.05
)
```

```
#> TOST results:
#> t-value lower bound: 0.553   p-value lower bound: 0.291
#> t-value upper bound: -3.32   p-value upper bound: 0.0008
#> degrees of freedom : 56.56
#>
#> Equivalence bounds (Cohen's d):
#> low eqbound: -0.5
#> high eqbound: 0.5
#>
#> Equivalence bounds (raw scores):
#> low eqbound: -0.4555
#> high eqbound: 0.4555
```

```
#>
#> TOST confidence interval:
#> lower bound 90% CI: -0.719
#> upper bound 90% CI:  0.068
#>
#> NHST confidence interval:
#> lower bound 95% CI: -0.797
#> upper bound 95% CI:  0.146
#>
#> Equivalence Test Result:
#> The equivalence test was non-significant, t(56.56) = 0.553, p = 0.291, given equivalence bound
#> Null Hypothesis Test Result:
#> The null hypothesis test was non-significant, t(56.56) = -1.384, p = 0.172, given an alpha of
```

- If both p-values are less than 0.05, we conclude that the groups are equivalent within the given range.

- The confidence interval helps visualize whether the effect size falls entirely within $[-0.5, 0.5]$.

16.2.4.1 Example 2: TOST for Correlation Equivalence

We can also use TOST to test whether a correlation coefficient is effectively zero.

```r
# Simulated correlation data
set.seed(123)
x <- rnorm(50)
y <- x * 0.02 + rnorm(50, sd = 1)  # Very weak correlation

# TOST for correlation
TOSTr(
    n = length(x),
    r = cor(x, y),
    low_eqbound_r = -0.1,
    high_eqbound_r = 0.1,
    alpha = 0.05
)
```

```
#> TOST results:
#> p-value lower bound: 0.280
#> p-value upper bound: 0.214
#>
#> Equivalence bounds (r):
#> low eqbound: -0.1
#> high eqbound: 0.1
#>
#> TOST confidence interval:
#> lower bound 90% CI: -0.25
#> upper bound 90% CI:  0.221
#>
#> NHST confidence interval:
#> lower bound 95% CI: -0.293
#> upper bound 95% CI:  0.264
#>
#> Equivalence Test Result:
#> The equivalence test was non-significant, p = 0.280, given equivalence bounds of -0
#> Null Hypothesis Test Result:
#> The null hypothesis test was non-significant, p = 0.915, given an alpha of 0.05.
```

- This tests whether the correlation is within $[-0.1, 0.1]$, meaning "practically zero".

- If both p-values are significant, we conclude that the correlation is effectively negligible.

16.2.5 Advantages of TOST Equivalence Testing

1. Avoids Misinterpretation of Non-Significance
   - Traditional NHST failing to reject H_0 does not imply equivalence.
   - TOST explicitly tests for equivalence, preventing misinterpretation.
2. Aligned with Confidence Intervals
   - TOST conclusions align with confidence interval-based reasoning.
3. Applicable to Various Statistical Tests
   - Can be used for means, correlations, regression coefficients, and more.
4. Commonly Used in Regulatory & Clinical Studies
   - Required for bioequivalence trials by organizations like the FDA (Schuirmann, 1987).

16.2.6 When *Not* to Use TOST

- If your research question is about detecting a difference rather than establishing equivalence.
- If the equivalence bounds are too wide to be meaningful in practice.
- If the sample size is too small, making it difficult to detect equivalence reliably.

Feature	Traditional NHST	TOST Equivalence Testing
Null Hypothesis	H_0: No effect ($\theta = 0$)	H_0: Effect is outside equivalence bounds
Alternative Hypothesis	H_a: There is an effect ($\theta \neq 0$)	H_a: Effect is within equivalence bounds
Goal	Detect difference	Establish similarity
p-value Interpretation	Small p means evidence for an effect	Small p means evidence for equivalence

16.3 False Discovery Rate

When conducting multiple hypothesis tests simultaneously, we increase the probability of false positives (Type I errors). Traditional correction methods like Bonferroni correction are too conservative, reducing statistical power.

The False Discovery Rate (FDR), introduced by Benjamini and Hochberg (1995), is a more flexible and powerful approach that controls the proportion of false discoveries (incorrect rejections of the null hypothesis) while maintaining a reasonable chance of detecting true effects.

Suppose we perform m independent hypothesis tests, each with a significance level α. The probability of making at least one Type I error (false positive) is:

$$P(\text{at least one false positive}) = 1 - (1 - \alpha)^m$$

For example, with $\alpha = 0.05$ and $m = 20$ tests:

$$P(\text{at least one false positive}) = 1 - (0.95)^{20} \approx 0.64$$

Thus, if we do not adjust for multiple testing, we are highly likely to reject at least one true null hypothesis just by chance.

Family-Wise Error Rate vs. False Discovery Rate

Approach	Controls	Method	Pros	Cons
Bonferroni Correction	FWER (Probability of ≥ 1 false positive)	Adjusts α: α/m	Very conservative, reduces false positives	Low power, increases false negatives
False Discovery Rate	Expected proportion of false discoveries	Adjusts p-values dynamically	Higher power, fewer false negatives	Some false positives allowed

Why FDR?

- FWER control (Bonferroni, Holm) is too strict, reducing true discoveries.
- FDR control allows a small fraction of false positives while keeping most discoveries valid.

Let:

- m = total number of hypotheses tested
- V = number of false discoveries (Type I errors)
- R = total number of rejected null hypotheses

Then, the False Discovery Rate is:

$$\text{FDR} = E\left[\frac{V}{\max(R, 1)}\right]$$

- If no null hypotheses are rejected ($R = 0$), we define FDR = 0.
- Unlike FWER, which controls the probability of any false positives, FDR controls the expected proportion of false positives.

16.3.1 Benjamini-Hochberg Procedure

The Benjamini-Hochberg (BH) procedure is the most widely used FDR-controlling method (Benjamini and Hochberg, 1995). It works as follows:

Step-by-Step Algorithm:

1. Perform m hypothesis tests and obtain p-values: $p_1, p_2, ..., p_m$.
2. Rank the p-values in ascending order: $p_{(1)} \leq p_{(2)} \leq ... \leq p_{(m)}$.
3. Calculate the Benjamini-Hochberg critical value for each test:

$$p_{(i)} \leq \frac{i}{m}\alpha$$

4. Find the largest i where $p_{(i)} \leq \frac{i}{m}\alpha$.
5. Reject all hypotheses with $p \leq p_{(i)}$.

Interpretation:

- This ensures that the expected proportion of false discoveries is controlled at level α.
- Unlike Bonferroni, it does not require independence of tests, making it more powerful.

16.3.1.1 Example 1: FDR Correction on Simulated Data

```r
set.seed(123)

# Generate 20 random p-values
p_values <-
    runif(20, 0, 0.1)  # Simulating p-values from multiple tests

# Apply FDR correction (Benjamini-Hochberg)
adjusted_p <- p.adjust(p_values, method = "BH")

# Compare raw and adjusted p-values
data.frame(Raw_p = round(p_values, 3),
           Adjusted_p = round(adjusted_p, 3)) |> head()
#>   Raw_p Adjusted_p
```

```
#> 1 0.029      0.095
#> 2 0.079      0.096
#> 3 0.041      0.095
#> 4 0.088      0.096
#> 5 0.094      0.096
#> 6 0.005      0.046
```

- Adjusted p-values control the expected proportion of false discoveries.
- If an adjusted p-value is below α, we reject the null hypothesis.

16.3.1.2 Example 2: FDR Correction in Gene Expression Analysis

FDR is widely used in genomics, where thousands of genes are tested for differential expression.

```r
library(multtest)

# Simulated gene expression study with 1000 genes
set.seed(42)
p_values <- runif(100, 0, 0.1)

# Apply different multiple testing corrections

# Bonferroni (very strict)
p_bonf <- p.adjust(p_values, method = "bonferroni")

# Holm's method
p_holm <- p.adjust(p_values, method = "holm")

# Benjamini-Hochberg (FDR)
p_fdr <- p.adjust(p_values, method = "BH")

# Compare significance rates
sum(p_bonf < 0.05)  # Strictest correction
#> [1] 3
sum(p_holm < 0.05)  # Moderately strict
#> [1] 3
sum(p_fdr < 0.05)   # Most discoveries, controlled FDR
#> [1] 5
```

- Bonferroni results in few discoveries (low power).
- Benjamini-Hochberg allows more discoveries, while controlling the proportion of false positives.

Use FDR when:

- You perform many hypothesis tests (e.g., genomics, finance, A/B testing).

- You want to balance false positives and false negatives.
- Bonferroni is too strict, leading to low power.

Do not use FDR if:

- Strict control of any false positives is required (e.g., drug approval studies).
- There are only a few tests, where Bonferroni is appropriate.

16.3.2 Benjamini-Yekutieli Procedure

The Benjamini-Yekutieli (BY) method modifies the Benjamini-Hochberg (BH) procedure to account for correlated test statistics (Benjamini and Yekutieli, 2001).

The key adjustment is a larger critical value for significance, making it more conservative than BH.

Similar to BH, the BY procedure ranks m p-values in ascending order:

$$p_{(1)} \leq p_{(2)} \leq \cdots \leq p_{(m)}$$

Instead of using $\alpha \frac{i}{m}$ as in BH, BY introduces a **correction factor**:

$$p_{(i)} \leq \frac{i}{mC(m)} \alpha$$

where:

$$C(m) = \sum_{j=1}^{m} \frac{1}{j} \approx \ln(m) + 0.577$$

- $C(m)$ is the harmonic series correction (ensures control under dependence).
- This makes the BY threshold larger than BH, reducing false positives.
- Recommended when tests are positively correlated (e.g., in fMRI, finance).

We can apply BY correction using the built-in `p.adjust()` function.

```
set.seed(123)

# Simulate 50 random p-values
p_values <- runif(50, 0, 0.1)

# Apply BY correction
```

```r
p_by <- p.adjust(p_values, method = "BY")

# Compare raw and adjusted p-values
data.frame(Raw_p = round(p_values, 3), Adjusted_BY = round(p_by, 3))
#>    Raw_p Adjusted_BY
#> 1  0.029       0.430
#> 2  0.079       0.442
#> 3  0.041       0.430
#> 4  0.088       0.442
#> 5  0.094       0.442
#> 6  0.005       0.342
#> 7  0.053       0.442
#> 8  0.089       0.442
#> 9  0.055       0.442
#> 10 0.046       0.430
#> 11 0.096       0.442
#> 12 0.045       0.430
#> 13 0.068       0.442
#> 14 0.057       0.442
#> 15 0.010       0.429
#> 16 0.090       0.442
#> 17 0.025       0.430
#> 18 0.004       0.342
#> 19 0.033       0.430
#> 20 0.095       0.442
#> 21 0.089       0.442
#> 22 0.069       0.442
#> 23 0.064       0.442
#> 24 0.099       0.447
#> 25 0.066       0.442
#> 26 0.071       0.442
#> 27 0.054       0.442
#> 28 0.059       0.442
#> 29 0.029       0.430
#> 30 0.015       0.429
#> 31 0.096       0.442
#> 32 0.090       0.442
#> 33 0.069       0.442
#> 34 0.080       0.442
#> 35 0.002       0.342
#> 36 0.048       0.430
#> 37 0.076       0.442
#> 38 0.022       0.430
#> 39 0.032       0.430
#> 40 0.023       0.430
```

```
#> 41 0.014        0.429
#> 42 0.041        0.430
#> 43 0.041        0.430
#> 44 0.037        0.430
#> 45 0.015        0.429
#> 46 0.014        0.429
#> 47 0.023        0.430
#> 48 0.047        0.430
#> 49 0.027        0.430
#> 50 0.086        0.442
```

Comparison: BH vs. BY

```
p_bh <- p.adjust(p_values, method = "BH")  # BH correction

# Visualize differences
data.frame(Raw_p = round(p_values, 3),
           BH_Adjusted = round(p_bh, 3),
           BY_Adjusted = round(p_by, 3)) |> head()
#>   Raw_p BH_Adjusted BY_Adjusted
#> 1 0.029       0.096       0.430
#> 2 0.079       0.098       0.442
#> 3 0.041       0.096       0.430
#> 4 0.088       0.098       0.442
#> 5 0.094       0.098       0.442
#> 6 0.005       0.076       0.342
```

- Benjamini-Yekutieli is more conservative than Benjamini-Hochberg.
- If BH identifies significant results but BY does not, the tests are likely correlated.

16.3.3 Storey's q-value Approach

Storey's q-value method directly estimates False Discovery Rate rather than adjusting individual p-values (Benjamini and Yekutieli, 2001).

Unlike BH/BY, it does not assume a fixed threshold (α) but estimates the FDR dynamically from the data.

Define:

- m_0: Number of **true null hypotheses**.
- π_0: Estimated **proportion of true nulls** in the dataset.

Storey's q-value adjusts p-values based on:

$$q(p) = \frac{\pi_0 \cdot m \cdot p}{\sum_{i=1}^{m} 1_{p_i \leq p}}$$

where:

- π_0 is estimated from the distribution of large p-values.
- Unlike BH/BY, Storey's method dynamically estimates the null proportion.

```r
# devtools::install_github("jdstorey/qvalue")
library(qvalue)

# Simulated data: 1000 hypothesis tests
set.seed(123)
p_values <- runif(1000, 0, 0.1)

# Compute q-values
qvals <- qvalue_truncp(p_values)$qvalues

# Summary of q-values
summary(qvals)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.3126  0.9689  0.9771  0.9684  0.9847  1.0000
```

Comparison: BH vs. BY vs. Storey

```r
# Apply multiple corrections
p_bh <- p.adjust(p_values, method = "BH")   # Benjamini-Hochberg
p_by <- p.adjust(p_values, method = "BY")   # Benjamini-Yekutieli
q_vals <- qvalue_truncp(p_values)$qvalues   # Storey's q-value

# Compare significance rates
data.frame(
  Raw_p = round(p_values[1:10], 3),
  BH = round(p_bh[1:10], 3),
  BY = round(p_by[1:10], 3),
  q_value = round(q_vals[1:10], 3)
)
#>    Raw_p    BH    BY q_value
#> 1  0.029 0.097 0.725   0.969
#> 2  0.079 0.098 0.737   0.985
#> 3  0.041 0.097 0.725   0.969
#> 4  0.088 0.099 0.740   0.989
#> 5  0.094 0.100 0.746   0.998
```

```
#> 6  0.005 0.094 0.700   0.936
#> 7  0.053 0.098 0.737   0.985
#> 8  0.089 0.099 0.740   0.989
#> 9  0.055 0.098 0.737   0.985
#> 10 0.046 0.098 0.731   0.977
```

16.3.4 Summary: False Discovery Rate Methods

FDR control methods balance Type I and Type II errors, making them more powerful than conservative Family-Wise Error Rate (FWER) methods like Bonferroni.

Method	Type	Strength	Best for
Benjamini-Hochberg	Adjusted p-values	Most powerful	Independent tests (e.g., surveys, psychology)
Benjamini-Yekutieli	Adjusted p-values	More conservative	Correlated tests (e.g., fMRI, finance)
Storey's q-value	Direct FDR estimation	Most flexible	Large-scale studies (e.g., genomics, proteomics)
Bonferroni	Family-Wise Error Rate (FWER)	Very conservative	Small number of tests, strict control
Holm's Method	FWER (less strict than Bonferroni)	Moderate	Moderately strict correction

---

16.4 Comparison of Testing Frameworks

Framework	Key Concept	Strengths	Weaknesses
NHST	Tests if an effect is statistically significant (p-value based)	Simple, widely used	Over-reliance on p-values, arbitrary thresholds (e.g., 0.05)

Framework	Key Concept	Strengths	Weaknesses
TOST	Tests whether two means are sufficiently close (equivalence)	Useful in equivalence testing	Requires pre-specified equivalence margin
Bayesian Testing	Uses posterior probabilities and Bayes Factor	Incorporates prior knowledge, intuitive	Requires prior distribution, computationally expensive
Decision-Theoretic	Minimizes expected loss based on cost functions	Practical for decision-making	Needs subjective cost assignment
False Discovery Rate	Controls proportion of false positives in multiple tests	Useful in high-dimensional data	Can still lead to false discoveries

Chapter 17

Marginal Effects

Marginal effects play a fundamental role in **interpreting regression models**, particularly when analyzing the impact of explanatory variables on an outcome variable. These effects provide a precise measure of **how a small change in an independent variable influences the dependent variable**.

The concept of a marginal effect is closely linked to **derivatives** in calculus. In simple linear models, marginal effects correspond directly to the estimated regression coefficients. However, in nonlinear models, computing marginal effects requires careful consideration, often involving either analytical differentiation or numerical approximation.

17.1 Definition of Marginal Effects

Mathematically, the **marginal effect** of an independent variable X on the expected value of a dependent variable Y is given by:

$$\frac{\partial E[Y|X]}{\partial X}$$

which represents the instantaneous rate of change of $E[Y|X]$ with respect to X.

For a **linear regression model**:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \varepsilon$$

the marginal effect of X_j is simply β_j. However, in more complex cases, such as nonlinear models, interaction effects, or transformations, marginal effects are not directly given by the regression coefficients and must be computed explicitly.

17.1.1 Analytical Derivation of Marginal Effects

In models where $E[Y|X]$ is a differentiable function of X, marginal effects are computed using calculus. The derivative of a function $f(x)$ is given by:

$$f'(x) \equiv \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Example: Quadratic Function

Consider the function:

$$f(x) = x^2.$$

The marginal effect is derived as follows:

$$\begin{aligned} f'(x) &= \lim_{h \to 0} \frac{(x+h)^2 - x^2}{h} \\ &= \frac{x^2 + 2xh + h^2 - x^2}{h} \\ &= \frac{2xh + h^2}{h} \\ &= 2x + h. \end{aligned}$$

As $h \to 0$, the marginal effect simplifies to:

$$f'(x) = 2x.$$

Thus, for small changes in x, the effect on $f(x)$ depends on x itself.

17.1.2 Numerical Approximation of Marginal Effects

In practice, analytical differentiation may be infeasible, particularly when dealing with **complex functions, large datasets, or models without closed-form derivatives**. In such cases, numerical differentiation provides an alternative.

17.1.2.1 One-Sided Numerical Approximation

A simple way to approximate the derivative is the forward difference formula:

$$
\begin{aligned}
f'(x) &= \lim_{h \to 0} \frac{(x+h)^2 - x^2}{h} \\
&\approx \frac{f(x+h) - f(x)}{h}
\end{aligned}
$$

where h is a small step size.

17.1.2.2 Two-Sided Numerical Approximation

A more accurate method is the central difference formula:

$$
f_2'(x) \approx \frac{f(x+h) - f(x-h)}{2h}.
$$

This approach reduces numerical error and is generally preferred in computational implementations.

17.1.2.3 Choosing an Appropriate h

The choice of h is critical (Gould et al., 2010, chapter 1):

- **Too small**: Can lead to numerical instability due to floating-point precision limitations.
- **Too large**: Reduces the accuracy of the approximation.

A common heuristic is to set $h = 10^{-5}$ or a small fraction of the standard deviation of X.

Comparison of Analytical and Numerical Methods

Table 17.1: Numerical derivatives are often preferred in empirical applications, especially when working with **complex models or machine learning algorithms**.

Method	Advantages	Disadvantages
Analytical	Provides exact expressions	Requires differentiability, not always feasible
Numerical	Works for any function, easy to implement	Requires careful choice of step size h

	Analytical Derivation	Numerical Approximation
Marginal Effects	Uses calculus (rules of differentiation)	Uses finite differences to approximate derivatives
Standard Errors	Derived using variance rules	Estimated via the delta method using the variance-covariance matrix

---

17.2 Marginal Effects in Different Contexts

1. **Linear Regression Models**

For a simple linear regression:

$$E[Y|X] = \beta_0 + \beta_1 X,$$

the marginal effect is **constant** and equal to β_1. This makes interpretation straightforward.

2. **Logit and Probit Models**

In logistic regression, the expected value of Y is modeled as:

$$E[Y|X] = P(Y = 1|X) = \frac{1}{1 + e^{-\beta_0 - \beta_1 X}}.$$

The marginal effect is given by:

$$\frac{\partial E[Y|X]}{\partial X} = \beta_1 P(Y = 1|X)(1 - P(Y = 1|X)).$$

Unlike linear models, the effect **varies with** X, requiring evaluation at specific values (e.g., means or percentiles).

3. **Interaction Effects and Nonlinear Terms**

When models include interactions (e.g., $X_1 X_2$) or transformations (e.g., $\log(X)$), marginal effects become more complex. For example, in:

$$E[Y|X] = \beta_0 + \beta_1 X + \beta_2 X^2,$$

the marginal effect of X is:

$$\frac{\partial E[Y|X]}{\partial X} = \beta_1 + 2\beta_2 X.$$

This means the marginal effect **depends on the value of** X.

17.3 Marginal Effects Interpretation

The interpretation of marginal effects differs depending on whether X is continuous or discrete:

Variable Type	Definition of Marginal Effect
Continuous	The derivative $\frac{\partial E[Y\|X]}{\partial X}$ represents an **infinitesimal** change in X.
Discrete	The change in $E[Y\|X]$ when X increases by one unit (also called an **incremental effect**).

For example, in a **binary variable case** (e.g., a dummy variable for gender), the marginal effect is:

$$E[Y|X = 1] - E[Y|X = 0].$$

which quantifies the expected change in Y when switching from $X = 0$ to $X = 1$.

17.4 Delta Method

The Delta Method is a statistical technique for approximating the **mean and variance** of a function of random variables. It is particularly useful in regression analysis when estimating the standard errors of **nonlinear functions** of estimated coefficients, such as:

- **Marginal effects in nonlinear models** (e.g., logistic regression)
- **Elasticities and risk measures** (e.g., in finance)
- **Transformation of regression coefficients** (e.g., log transformations)

This method is based on a first-order Taylor Series approximation, which allows us to estimate the variance of a transformed parameter without requiring explicit distributional assumptions.

Let $G(\beta)$ be a function of the estimated parameters β, where β follows an asymptotically normal distribution:

$$\beta \sim N(\hat{\beta}, \mathrm{Var}(\hat{\beta})).$$

Using a **first-order Taylor expansion**, we approximate $G(\beta)$ around its expectation:

$$G(\beta) \approx G(\hat{\beta}) + \nabla G(\beta)(\beta - \hat{\beta}),$$

where $\nabla G(\beta)$ is the gradient (also known as the **Jacobian**) of $G(\beta)$, i.e., the vector of partial derivatives:

$$\nabla G(\beta) = \left(\frac{\partial G}{\partial \beta_1}, \frac{\partial G}{\partial \beta_2}, \dots, \frac{\partial G}{\partial \beta_k} \right).$$

The variance of $G(\beta)$ is then approximated as:

$$\mathrm{Var}(G(\beta)) \approx \nabla G(\beta)\, \mathrm{Cov}(\beta)\, \nabla G(\beta)'.$$

where:

- $\nabla G(\beta)$ is the gradient vector of $G(\beta)$.
- $\mathrm{Cov}(\beta)$ is the variance-covariance matrix of $\hat{\beta}$.
- The expression $\nabla G(\beta)'$ denotes the transpose of the gradient.

Key Properties of the Delta Method

- **Semi-parametric approach**: It does not require full knowledge of the distribution of $G(\beta)$.
- **Widely applicable**: Useful for computing standard errors in regression models.
- **Alternative approaches**:
  - **Analytical derivation**: Directly deriving a probability function for the margin.
  - **Simulation/Bootstrapping**: Using Monte Carlo methods to approximate standard errors.

17.5 Comparison: Delta Method vs. Alternative Approaches

Method	Description	Pros	Cons
Delta Method	Uses Taylor expansion to approximate variance	Computationally efficient	Accuracy depends on linearity assumption

Method	Description	Pros	Cons
Analytical Derivation	Directly derives probability function	Exact solution (if feasible)	Can be mathematically complex
Simulation/Bootstrapping	Uses repeated sampling from estimated distribution	No assumptions on functional form	Computationally expensive

When to Use the Delta Method:

- When you need a **quick approximation** for standard errors.
- When the function $G(\beta)$ is **smooth and differentiable**.
- When working with **large sample sizes**, where asymptotic normality holds.

For deeper exploration, refer to these excellent resources:

- **Advanced**: modmarg package documentation – covers implementation of the Delta Method in R.
- **Intermediate**: UCLA Statistical Consulting – a practical FAQ on the Delta Method.

17.5.1 Example: Applying the Delta Method in a logistic regression

To illustrate, let's apply the **Delta Method** to compute the standard error of a nonlinear transformation of regression coefficients.

In logistic regression, the estimated coefficient $\hat{\beta}$ represents the **log-odds** change for a one-unit increase in X. However, we often want the **odds ratio**, which is:

$$G(\beta) = e^{\beta}.$$

By the **Delta Method**, the variance of e^{β} is:

$$\text{Var}(e^{\beta}) \approx e^{2\beta} \cdot \text{Var}(\beta).$$

```r
# Load necessary packages
library(ggplot2)
library(margins)
library(sandwich)
```

```r
library(lmtest)

# Simulate data
set.seed(123)
n <- 100
X <- rnorm(n)  # Simulate independent variable
# Generate binary outcome using logistic model
Y <-
    rbinom(n, 1, plogis(0.5 + 0.8 * X))

# Logistic regression
logit_model <- glm(Y ~ X, family = binomial(link = "logit"))

# Extract coefficient and variance
beta_hat <- coef(logit_model)["X"]   # Estimated coefficient
var_beta_hat <- vcov(logit_model)["X", "X"]  # Variance of beta_hat

# Apply Delta Method
odds_ratio <- exp(beta_hat)  # Transform beta to odds ratio
se_odds_ratio <-
    sqrt(odds_ratio ^ 2 * var_beta_hat)  # Delta Method SE

# Compute 95% Confidence Interval
lower_CI <- exp(beta_hat - 1.96 * sqrt(var_beta_hat))
upper_CI <- exp(beta_hat + 1.96 * sqrt(var_beta_hat))

# Display results
results <- data.frame(
    Term = "X",
    Odds_Ratio = odds_ratio,
    SE = se_odds_ratio,
    Lower_CI = lower_CI,
    Upper_CI = upper_CI
)

print(results)
#>   Term Odds_Ratio        SE Lower_CI Upper_CI
#> X    X   2.655431 0.7677799 1.506669 4.680069

# ---- VISUALIZATION 1: Distribution of Simulated Odds Ratios ----
set.seed(123)
# Simulate beta estimates
simulated_betas <-
    rnorm(1000, mean = beta_hat, sd = sqrt(var_beta_hat))
simulated_odds_ratios <-
```

```
    exp(simulated_betas)  # Apply transformation

ggplot(data.frame(Odds_Ratio = simulated_odds_ratios),
       aes(x = Odds_Ratio)) +
    geom_histogram(
        color = "black",
        fill = "skyblue",
        bins = 50,
        alpha = 0.7
    ) +
    geom_vline(
        xintercept = odds_ratio,
        color = "red",
        linetype = "dashed",
        linewidth = 1.2
    ) +
    geom_vline(
        xintercept = lower_CI,
        color = "blue",
        linetype = "dotted",
        linewidth = 1.2
    ) +
    geom_vline(
        xintercept = upper_CI,
        color = "blue",
        linetype = "dotted",
        linewidth = 1.2
    ) +
    labs(title = "Distribution of Simulated Odds Ratios",
         x = "Odds Ratio",
         y = "Frequency") +
    theme_minimal()
```

- **Odds Ratio Computation**
  - The odds ratio is the exponentiated coefficient $e^{\hat{\beta}}$, which represents the multiplicative change in the odds of $Y = 1$ for a one-unit increase in X.
  - If $\hat{\beta} = 0.8$, then $e^{0.8} \approx 2.23$, meaning a one-unit increase in X increases the odds of $Y = 1$ by **123%**.
- **Standard Error via Delta Method**
  - Since β follows a normal distribution, its transformation e^{β} is **not normally distributed** but rather follows a log-normal shape.
  - The Delta Method approximates the **standard error** of e^{β} using:

$$SE(e^{\beta}) = \sqrt{e^{2\beta} \cdot \text{Var}(\beta)}$$

- **Confidence Intervals**
  - The confidence interval is obtained by:

$$[e^{\beta - 1.96 \cdot SE}, e^{\beta + 1.96 \cdot SE}]$$

  - This helps interpret the uncertainty around the odds ratio estimate.
- **Visualization**
  - The **histogram of simulated odds ratios** shows how the transformation affects variance:
    - The **red dashed line** represents the estimated odds ratio.

* The **blue dotted lines** show the **confidence interval bounds**.
* The **right-skewed distribution** reflects the non-linear transformation, meaning higher uncertainty for larger values.

17.6 Types of Marginal Effect

17.6.1 Average Marginal Effect

The Average Marginal Effect (AME) measures the **expected change in the predicted probability** when an independent variable **increases by a small amount** while holding all other variables constant. Unlike marginal effects at the mean (MEM), AMEs average the marginal effects **across all observations**, providing a more representative measure.

Applications of AMEs

- **Marketing**: How much does increasing ad spend change the probability of a customer purchase?
- **Finance**: How does an interest rate change impact the probability of loan approval?
- **Econometrics**: What is the effect of education on the probability of employment?

Since nonlinear models like **logit** and **probit** do not have constant marginal effects, AMEs require numerical differentiation. There are two common approaches:

1. One-Sided Numerical Derivative: Uses a small forward step h to estimate the derivative.
2. Two-Sided Numerical Derivative: Takes both a forward step and a backward step to improve accuracy.

17.6.1.1 One-Sided Numerical Derivative

To estimate $\frac{\partial p(\mathbf{X},\beta)}{\partial X}$ numerically:

Algorithm

1. Estimate the model using logistic (or probit) regression.
2. For each observation i:
    1. Compute predicted probability using the observed data:
    $$\widehat{Y}_{i0} = p(\mathbf{X}_i, \widehat{\beta}).$$
    2. Increase X by a small step h, where:
        - If X is continuous, choose:
    $$h = (|\bar{X}| + 0.001) \times 0.001.$$

- If X is discrete, set $h = 1$.

3. Compute the new predicted probability:

$$\hat{Y}_{i1} = p(\mathbf{X}_i + h, \hat{\beta}).$$

4. Compute the numerical derivative:

$$\frac{\hat{Y}_{i1} - \hat{Y}_{i0}}{h}.$$

3. Average across all observations:

$$E\left[\frac{\hat{Y}_{i1} - \hat{Y}_{i0}}{h}\right] \approx \frac{\partial p(Y|\mathbf{X}, \beta)}{\partial X}.$$

```r
# Load necessary packages
library(margins)
library(sandwich)
library(lmtest)

# Simulate data
set.seed(123)
n <- 100
X <- rnorm(n)
Y <- rbinom(n, 1, plogis(0.5 + 0.8 * X))  # Logistic function

# Logistic regression
logit_model <- glm(Y ~ X, family = binomial(link = "logit"))

# Define step size h for continuous variable
X_mean <- mean(X)
h <- (abs(X_mean) + 0.001) * 0.001

# Compute predicted probabilities at original X
pred_Y0 <- predict(logit_model, type = "response")

# Compute predicted probabilities at X + h
X_new <- X + h
data_new <- data.frame(X = X_new)
pred_Y1 <-
    predict(logit_model, newdata = data_new, type = "response")

# Compute marginal effects
marginal_effects <- (pred_Y1 - pred_Y0) / h

# Compute Average Marginal Effect (AME)
AME_one_sided <- mean(marginal_effects)
```

```r
# Display results
data.frame(Method = "One-Sided AME", Estimate = AME_one_sided)
#>          Method  Estimate
#> 1 One-Sided AME 0.1921614
```

- The AME is the average effect of X on the probability of $Y = 1$.
- Since logistic regression is nonlinear, the effect varies across observations.
- This method assumes a small h provides a good approximation.

17.6.1.2 Two-Sided Numerical Derivative

To improve accuracy, we use the **two-sided derivative**:

Algorithm

1. Estimate the model using logistic (or probit) regression.
2. For each observation i:
   1. Compute the original predicted probability:
   $$\hat{Y}_{i0} = p(\mathbf{X}_i, \hat{\beta}).$$
   2. Compute the new predicted probabilities:
      - Increase X by h:
      $$\hat{Y}_{i1} = p(\mathbf{X}_i + h, \hat{\beta}).$$
      - Decrease X by h:
      $$\hat{Y}_{i2} = p(\mathbf{X}_i - h, \hat{\beta}).$$
   3. Compute the numerical derivative:
   $$\frac{\hat{Y}_{i1} - \hat{Y}_{i2}}{2h}.$$
3. Average across all observations:
$$E\left[\frac{\hat{Y}_{i1} - \hat{Y}_{i2}}{2h}\right] \approx \frac{\partial p(Y|\mathbf{X}, \beta)}{\partial X}.$$

```r
# Compute predicted probabilities at X - h
X_new_minus <- X - h
data_new_minus <- data.frame(X = X_new_minus)
pred_Y2 <-
    predict(logit_model, newdata = data_new_minus, type = "response")
```

```r
# Compute two-sided marginal effects
marginal_effects_2sided <- (pred_Y1 - pred_Y2) / (2 * h)

# Compute Average Marginal Effect (AME) - Two-Sided
AME_two_sided <- mean(marginal_effects_2sided)

# Display results
data.frame(Method = "Two-Sided AME", Estimate = AME_two_sided)
#>          Method  Estimate
#> 1 Two-Sided AME 0.1921633
```

Comparison of One-Sided vs. Two-Sided AME

Method	Accuracy	Computational Cost	Bias
One-Sided	Lower	Faster	Higher
Two-Sided	Higher	Slightly Slower	Lower

- **One-sided** AME is computationally simpler but can introduce bias.
- **Two-sided** AME reduces bias but requires two function evaluations per observation.

17.6.2 Marginal Effects at the Mean

Marginal effects in nonlinear models are not constant, as they depend on the values of independent variables. One way to summarize them is by computing Marginal Effects at the Mean (MEM), which estimates marginal effects **at the average values of the independent variables**.

MEM is commonly used in:

- **Econometrics**: Evaluating the effect of education on wages at the average level of experience.
- **Finance**: Assessing the impact of credit scores on loan approval probability for a typical applicant.
- **Marketing**: Estimating the effect of price on purchase probability for an average customer.

Unlike the Average Marginal Effect, which averages marginal effects over all observations, MEM computes the effect **at a single point**—the mean of the explanatory variables.

Let $p(\mathbf{X}, \beta)$ be the predicted probability in a nonlinear model (e.g., logistic regression). The **MEM** is computed as:

$$\frac{\partial p(\bar{\mathbf{X}}, \beta)}{\partial X}$$

where $\bar{\mathbf{X}}$ is the **vector of mean values** of all explanatory variables.

For a logistic regression model:

$$E[Y|X] = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

the MEM for a continuous variable X is:

$$\left.\frac{\partial E[Y|X]}{\partial X}\right|_{X=\bar{X}} = \beta_1 \cdot p(\bar{X}) \cdot (1 - p(\bar{X})).$$

MEM vs. AME

Method	Description	Pros	Cons
MEM	Marginal effect computed at the mean values of predictors.	Easy to interpret, based on a single reference point.	Not representative if data is skewed or nonlinear.
AME	Average of marginal effects across all observations.	More generalizable, considers full distribution of data.	Computationally more expensive.

Step 1: Estimate the Model

```r
# Load necessary packages
library(margins)
library(sandwich)
library(lmtest)

# Simulate data
set.seed(123)
n <- 100
X <- rnorm(n)
Y <- rbinom(n, 1, plogis(0.5 + 0.8 * X))  # Logistic function

# Logistic regression
logit_model <- glm(Y ~ X, family = binomial(link = "logit"))
```

Step 2: Compute Marginal Effects at the Mean

```r
# Compute mean of X
X_mean <- mean(X)

# Compute predicted probability at mean X
p_mean <-
    predict(logit_model,
            newdata = data.frame(X = X_mean),
            type = "response")

# Compute MEM for X
MEM <- coef(logit_model)["X"] * p_mean * (1 - p_mean)

# Display result
data.frame(Method = "MEM", Estimate = MEM)
#>   Method  Estimate
#> X    MEM 0.2146628
```

Interpretation

- The MEM tells us the effect of X on the probability of $Y = 1$ at the mean value of X.
- It provides a simple interpretation but may not capture the variability in marginal effects across different values of X.

A third approach is **Marginal Effects at Representative Values (MER)**, where we calculate marginal effects at **specific percentiles** (e.g., median, quartiles). Below is a comparison:

Method	Description	Pros	Cons
MEM	Marginal effect at mean values of predictors.	Simple to compute, useful if data is symmetric.	Not informative if mean is not representative.
AME	Average of marginal effects over all observations.	More generalizable.	More computationally expensive.
MER	Marginal effect at specific values (e.g., median, percentiles).	Captures effects at different levels of X.	Requires choosing relevant reference values.

When to Use Each Method

- Use **MEM** when you need a quick, interpretable summary at an "average" individual.
- Use **AME** when marginal effects **vary widely across individuals**.

- Use **MER** when you need to understand effects at **specific values of interest**.

17.6.3 Marginal Effects at the Average

Marginal effects summarize how an independent variable influences the probability of an outcome in **nonlinear models** (e.g., logistic regression). We have already discussed:

- Marginal Effects at the Mean (MEM): Marginal effects computed at the mean values of the independent variables.
- Average Marginal Effects (AME): The mean of marginal effects computed at each observation.

A third approach is Marginal Effects at the Average(MAE), where we **first average the independent variables** across all observations, and then compute the marginal effect at that single averaged observation.

Let $p(\mathbf{X}, \beta)$ be the probability function of a model (e.g., a logistic regression). The Marginal Effect at the Average is computed as:

$$\frac{\partial p(\bar{\mathbf{X}}, \beta)}{\partial X}$$

where $\bar{\mathbf{X}}$ is the vector of **averaged independent variables across all observations**.

Key Differences Between AME and MAE

- AME answers a **general** question: "How does X affect Y across the entire dataset?"
- MAE answers a more **specific** question: "How does X affect Y for a typical (average) person in our dataset?"

MAE is particularly relevant when we want a single, interpretable effect for a representative individual.

Use Cases for MAE

1. **Policy & Business Decision-Making**: If policymakers or business leaders want to know the effect of a tax increase on a "typical" consumer, MAE gives an effect for a single representative individual.
2. **Marketing Campaigns**: If a marketing team wants to know how much increasing ad spend affects the purchase probability of an "average" customer, MAE provides this insight.

3. **Simplified Reporting**: AMEs vary across individuals, which can make reporting complex. MAE condenses everything into one easy-to-interpret number.

Comparison: MAE vs. MEM vs. AME

Method	Definition	Pros	Cons
MEM	Compute marginal effects **at the mean values** of X.	Simple and interpretable.	Mean values may not represent actual observations.
AME	Compute marginal effects **for each observation**, then take the average.	More robust, accounts for variability.	Computationally more expensive.
MAE	Compute probability at **averaged** X values, then compute the marginal effect.	Accounts for interactions better than MEM.	Less commonly used, can be misleading if X values are skewed.

Intuition Behind MAE

- Instead of computing individual marginal effects (as in AME), MAE computes the **marginal effect for a single averaged observation**.
- This method is **somewhat similar to MEM**, but instead of taking the mean of each independent variable separately, it **first computes a single averaged observation and then derives the marginal effect at that observation**.

Step 1: Estimate the Model

```r
# Load necessary packages
library(margins)

# Simulate data
set.seed(123)
n <- 100
X1 <- rnorm(n)  # Continuous variable
X2 <- rbinom(n, 1, 0.5)  # Binary variable
# Logistic function
Y <-
    rbinom(n, 1, plogis(0.5 + 0.8 * X1 - 0.5 * X2))

# Logistic regression
logit_model <- glm(Y ~ X1 + X2, family = binomial(link = "logit"))
```

Step 2: Compute MAE

```
# Compute the average of independent variables
X_mean <- data.frame(X1 = mean(X1), X2 = mean(X2))

# Compute predicted probability at averaged X
p_mean <- predict(logit_model, newdata = X_mean, type = "response")

# Compute MAE for X1
MAE_X1 <- coef(logit_model)["X1"] * p_mean * (1 - p_mean)

# Compute MAE for X2
MAE_X2 <- coef(logit_model)["X2"] * p_mean * (1 - p_mean)

# Display results
data.frame(
    Method = "MAE",
    Variable = c("X1", "X2"),
    Estimate = c(MAE_X1, MAE_X2)
)
#>    Method Variable    Estimate
#> X1    MAE       X1  0.20280618
#> X2    MAE       X2 -0.06286593
```

- The MAE for X_1 represents the change in probability when increasing X_1 at the average values of X_1 and X_2.

- The MAE for X_2 (a binary variable) represents the probability change when switching from $X_2 = 0$ to $X_2 = 1$, holding all other variables at their average.

Method	Computes Marginal Effect...	Accounts for Variability?	Best for...
MEM	At the mean of each independent variable.	No	Quick interpretation at a reference point. When individual means are meaningful (e.g., symmetric data).
AME	At each observation, then averages.	Yes	Generalizable results.

MAE	At a single averaged observation.	No	Simple summary when interactions exist. When we need a single interpretable summary that accounts for interactions

When to Use MAE

- When you want a single number summary that reflects a realistic scenario.
- When there are interaction effects, and you want to account for the joint impact of predictors.
- However, if predictor distributions are skewed, AME is usually preferred.

17.7 Packages for Marginal Effects

Several R packages compute **marginal effects** for regression models, each with different features and functionalities. The primary packages include:

1. `marginaleffects` – A modern, flexible, and efficient package.
2. `margins` – A widely used package for replicating Stata's `margins` command.
3. `mfx` – A package tailored for Generalized Linear Models (`glm`).

These tools help analyze how small changes in explanatory variables impact the dependent variable.

17.7.1 `marginaleffects` Package (Recommended)

The `marginaleffects` package is the successor to `margins` and `emtrends`, offering a faster, more efficient, and more flexible approach to estimating marginal effects.

Why Use `marginaleffects`?

- Supports interaction effects and complex models
- Computes marginal effects, marginal means, and counterfactuals
- Integrates well with ggplot2 for visualization
- Works with many model types (linear, logistic, Poisson, etc.)

Limitation:

- No built-in function for **multiple comparisons correction**, but you can use `p.adjust()` for adjustment.

Key Definitions

- **Marginal Effects**: The partial derivative (slope) of the outcome with respect to each variable.
  - `margins` package defines marginal effects as "partial derivatives of the regression equation with respect to each variable in the model for each unit in the data."
- **Marginal Means**: The expected outcome averaged over a grid of predictor values.

Computing Predictions and Marginal Effects

```
library(marginaleffects)
library(tidyverse)
data(mtcars)


# Fit a regression model with interaction terms
mod <- lm(mpg ~ hp * wt * am, data = mtcars)

# Get predicted values
predictions(mod) %>% head()
#>
#>  Estimate Std. Error    z Pr(>|z|)     S 2.5 % 97.5 %
#>      22.5      0.884 25.4   <0.001 471.7  20.8   24.2
#>      20.8      1.194 17.4   <0.001 223.3  18.5   23.1
#>      25.3      0.709 35.7   <0.001 922.7  23.9   26.7
#>      20.3      0.704 28.8   <0.001 601.5  18.9   21.6
#>      17.0      0.712 23.9   <0.001 416.2  15.6   18.4
#>      19.7      0.875 22.5   <0.001 368.8  17.9   21.4
#>
#> Columns: rowid, estimate, std.error, statistic, p.value, s.value, conf.low, conf.high, mpg, hp
#> Type:  response

# Create a reference grid for prediction
newdata <- datagrid(am = 0,
                    wt = c(2, 4),
                    model = mod)

# Plot predictions for 'hp' and 'wt'
marginaleffects::plot_predictions(mod,
                                  newdata = newdata,
                                  condition = c("hp", "wt"))
```

Computing Marginal Effects

```
# Compute Average Marginal Effects (AME)
mfx <- marginaleffects::slopes(mod, variables = c("hp", "wt"))
head(mfx)
#>
#>  Term Estimate Std. Error     z Pr(>|z|)   S   2.5 %    97.5 %
#>    hp  -0.0369     0.0185 -1.99  0.04605 4.4 -0.0732 -0.000648
#>    hp  -0.0287     0.0156 -1.84  0.06626 3.9 -0.0593  0.001926
#>    hp  -0.0466     0.0226 -2.06  0.03926 4.7 -0.0909 -0.002293
#>    hp  -0.0423     0.0133 -3.18  0.00146 9.4 -0.0683 -0.016232
#>    hp  -0.0390     0.0134 -2.91  0.00363 8.1 -0.0653 -0.012730
#>    hp  -0.0387     0.0135 -2.87  0.00409 7.9 -0.0652 -0.012289
#>
#> Columns: rowid, term, estimate, std.error, statistic, p.value, s.value, conf.low, c
#> Type:  response

# Compute Group-Average Marginal Effects
head(marginaleffects::slopes(mod, by = "hp", variables = "am"))
#>
#>  Term          Contrast hp Estimate Std. Error      z Pr(>|z|)   S 2.5 % 97.5 %
#>    am mean(1) - mean(0) 52     3.98       5.20  0.764    0.445 1.2 -6.22  14.18
#>    am mean(1) - mean(0) 62    -2.77       2.51 -1.107    0.268 1.9 -7.68   2.14
#>    am mean(1) - mean(0) 65     3.00       4.13  0.725    0.468 1.1 -5.10  11.10
#>    am mean(1) - mean(0) 66     2.03       3.48  0.582    0.561 0.8 -4.80   8.85
#>    am mean(1) - mean(0) 91     1.86       2.76  0.674    0.500 1.0 -3.54   7.26
#>    am mean(1) - mean(0) 93     1.20       2.35  0.511    0.609 0.7 -3.40   5.80
```

```
#>
#> Columns: term, contrast, hp, estimate, std.error, statistic, p.value, s.value, conf.low, conf.
#> Type:  response

# Marginal Effects at Representative Values (MER)
marginaleffects::slopes(mod, newdata = datagrid(am = 0, wt = c(2, 4)))
#>
#>  Term Contrast am wt Estimate Std. Error      z Pr(>|z|)   S   2.5 %   97.5 %
#>    am    1 - 0  0  2   2.5465     2.7860  0.914   0.3607 1.5 -2.9139  8.00694
#>    am    1 - 0  0  4  -2.9661     3.0381 -0.976   0.3289 1.6 -8.9207  2.98852
#>    hp    dY/dX  0  2  -0.0598     0.0283 -2.115   0.0344 4.9 -0.1153 -0.00439
#>    hp    dY/dX  0  4  -0.0309     0.0187 -1.654   0.0981 3.4 -0.0676  0.00571
#>    wt    dY/dX  0  2  -2.6762     1.4199 -1.885   0.0595 4.1 -5.4591  0.10676
#>    wt    dY/dX  0  4  -2.6762     1.4206 -1.884   0.0596 4.1 -5.4605  0.10816
#>
#> Columns: rowid, term, contrast, estimate, std.error, statistic, p.value, s.value, conf.low, co
#> Type:  response

# Marginal Effects at the Mean (MEM)
marginaleffects::slopes(mod, newdata = "mean")
#>
#>  Term Contrast Estimate Std. Error      z Pr(>|z|)   S   2.5 %  97.5 %
#>    am    1 - 0  -0.8086     1.5238 -0.531  0.59568 0.7 -3.7952  2.1781
#>    hp    dY/dX  -0.0422     0.0133 -3.181  0.00147 9.4 -0.0683 -0.0162
#>    wt    dY/dX  -2.6762     1.4193 -1.886  0.05935 4.1 -5.4579  0.1055
#>
#> Columns: rowid, term, contrast, estimate, std.error, statistic, p.value, s.value, conf.low, co
#> Type:  response
```

Counterfactual Comparisons

```
# Counterfactual comparison: Effect of changing 'am' from 0 to 1
comparisons(mod, variables = list(am = 0:1))
#>
#>  Term Contrast Estimate Std. Error      z Pr(>|z|)   S 2.5 % 97.5 %
#>    am    1 - 0    0.325       1.68  0.193    0.847 0.2 -2.97   3.62
#>    am    1 - 0   -0.544       1.57 -0.347    0.729 0.5 -3.62   2.53
#>    am    1 - 0    1.201       2.35  0.511    0.609 0.7 -3.40   5.80
#>    am    1 - 0   -1.703       1.87 -0.912    0.362 1.5 -5.36   1.96
#>    am    1 - 0   -0.615       1.68 -0.366    0.715 0.5 -3.91   2.68
#> --- 22 rows omitted. See ?avg_comparisons and ?print.marginaleffects ---
#>    am    1 - 0    4.081       3.94  1.037    0.300 1.7 -3.63  11.79
#>    am    1 - 0    2.106       2.29  0.920    0.358 1.5 -2.38   6.59
#>    am    1 - 0    0.895       1.64  0.544    0.586 0.8 -2.33   4.12
#>    am    1 - 0    4.027       3.24  1.243    0.214 2.2 -2.32  10.38
#>    am    1 - 0   -0.237       1.59 -0.149    0.881 0.2 -3.35   2.87
```

```
#> Columns: rowid, term, contrast, estimate, std.error, statistic, p.value, s.value, c
#> Type:  response
```

17.7.2 margins Package

The `margins` package is a popular choice, designed to replicate Stata's `margins` command in R. It provides marginal effects for each variable in a model, including interaction terms. `margins` define:

1. **Average Partial Effects:** The contribution of each variable to the outcome scale, conditional on the other variables involved in the link function transformation of the linear predictor.
2. **Average Marginal Effects:** The marginal contribution of each variable on the scale of the linear predictor.
3. Average marginal effects represent the mean of these unit-specific partial derivatives over a given sample.

Key Features

- Computes Average Partial Effects (APE).
- Supports Marginal Effects at the Mean (MEM).
- Provides visualizations of marginal effects.

Limitation:

- Slower than `marginaleffects`, especially with large datasets.

```r
library(margins)

# Fit a linear model with interactions
mod <- lm(mpg ~ cyl * hp + wt, data = mtcars)

# Compute marginal effects
summary(margins(mod))
#>  factor     AME     SE       z      p   lower   upper
#>     cyl  0.0381 0.5999  0.0636 0.9493 -1.1376  1.2139
#>      hp -0.0463 0.0145 -3.1909 0.0014 -0.0748 -0.0179
#>      wt -3.1198 0.6613 -4.7175 0.0000 -4.4160 -1.8236

# Equivalent function for summary
margins_summary(mod)
#>  factor     AME     SE       z      p   lower   upper
#>     cyl  0.0381 0.5999  0.0636 0.9493 -1.1376  1.2139
#>      hp -0.0463 0.0145 -3.1909 0.0014 -0.0748 -0.0179
#>      wt -3.1198 0.6613 -4.7175 0.0000 -4.4160 -1.8236
```

```r
# Plot marginal effects
plot(margins(mod))
```

Marginal Effects at Representative Values

```r
# Compute marginal effects when 'hp' = 150
margins(mod, at = list(hp = 150)) %>% summary()
#>  factor       hp     AME     SE       z      p   lower   upper
#>     cyl 150.0000  0.1009 0.6128  0.1647 0.8692 -1.1001  1.3019
#>      hp 150.0000 -0.0463 0.0145 -3.1909 0.0014 -0.0748 -0.0179
#>      wt 150.0000 -3.1198 0.6613 -4.7175 0.0000 -4.4160 -1.8236
```

17.7.3 `mfx` Package

The **`mfx`** package is specialized for GLMs, computing marginal effects for probit, logit, Poisson, and other count models.

Limitation:

- Computes only marginal effects for each variable, not the average marginal effect.

Supported Models in `mfx`

Model	Outcome Type	Function
Probit	Binary	`probitmfx()`
Logit	Binary	`logitmfx()`
Poisson	Count	`poissonmfx()`

Model	Outcome Type	Function
Negative Binomial	Count	`negbinmfx()`
Beta	Rate	`betamfx()`

Example: Poisson Regression

```
library(mfx)
data("mtcars")

# Fit a Poisson model and compute marginal effects
poissonmfx(formula = vs ~ mpg * cyl * disp, data = mtcars)
#> Call:
#> poissonmfx(formula = vs ~ mpg * cyl * disp, data = mtcars)
#>
#> Marginal Effects:
#>                     dF/dx   Std. Err.       z  P>|z|
#> mpg           1.4722e-03  8.7531e-03  0.1682 0.8664
#> cyl           6.6420e-03  3.9263e-02  0.1692 0.8657
#> disp          1.5899e-04  9.4555e-04  0.1681 0.8665
#> mpg:cyl      -3.4698e-04  2.0564e-03 -0.1687 0.8660
#> mpg:disp     -7.6794e-06  4.5545e-05 -0.1686 0.8661
#> cyl:disp     -3.3837e-05  1.9919e-04 -0.1699 0.8651
#> mpg:cyl:disp  1.6812e-06  9.8919e-06  0.1700 0.8650
```

For more details, see the `mfx` vignette.

17.7.4 Comparison of Packages

Feature	`marginaleffects` (Recommended)	`margins`	`mfx`
Computes AME	Yes	Yes	No
Computes MEM	Yes	Yes	No
Computes MER	Yes	Yes	No
Supports GLM Models	Yes	Yes	Yes (but limited to `glm`)
Works with Interactions	Yes	Yes	No
Fast and Efficient	Yes	No (slower)	Yes (for GLMs)
Supports Counterfactuals	Yes	No	No

Chapter 18

Moderation

Moderation analysis examines how the relationship between an independent variable (X) and a dependent variable (Y) changes depending on a third variable, the *moderator* (M). In regression terms, moderation is represented as an interaction effect.

18.1 Types of Moderation Analyses

There are two primary approaches to analyzing moderation:

1. Spotlight Analysis
   - Also known as **Simple Slopes Analysis**.
   - Examines the effect of X on Y at specific values of M (e.g., mean, ± 1 SD, percentiles).
   - Typically used for categorical or discretized moderators.
2. Floodlight Analysis
   - Extends spotlight analysis to examine moderation across the entire range of M.
   - Based on Johnson-Neyman Intervals, identifying values of M where the effect of X on Y is statistically significant.
   - Useful when the moderator is continuous and no specific cutoffs are predefined.

18.2 Key Terminology

- **Main Effect**: The effect of an independent variable without considering interactions.
- **Interaction Effect**: The combined effect of X and M on Y.

- **Simple Slope**: The slope of X on Y at a specific value of M (used when M is continuous).
- **Simple Effect**: The effect of X on Y at a particular level of M when X is categorical.

18.3 Moderation Model

A typical moderation model is represented as:

$$Y = \beta_0 + \beta_1 X + \beta_2 M + \beta_3 X \times M + \varepsilon$$

where:

- β_0: Intercept
- β_1: Main effect of X
- β_2: Main effect of M
- β_3: Interaction effect of X and M

If β_3 is significant, it suggests that the effect of X on Y depends on M.

18.4 Types of Interactions

1. **Continuous by Continuous**: Both X and M are continuous (e.g., age moderating the effect of income on spending).
2. **Continuous by Categorical**: X is continuous, and M is categorical (e.g., gender moderating the effect of education on salary).
3. **Categorical by Categorical**: Both X and M are categorical (e.g., the effect of a training program on performance, moderated by job role).

18.5 Three-Way Interactions

For models with a second moderator (W), we examine:

$$\begin{aligned} Y = \beta_0 &+ \beta_1 X + \beta_2 M + \beta_3 W + \beta_4 X \times M \\ &+ \beta_5 X \times W + \beta_6 M \times W + \beta_7 X \times M \times W + \varepsilon \end{aligned}$$

- To interpret three-way interactions, the **slope difference test** can be used (Dawson and Richter, 2006).

18.6 Additional Resources

- **Bayesian ANOVA models**: `BANOVAL` package allows floodlight analysis.

- **Structural Equation Modeling**: `cSEM` package includes `doFloodlightAnalysis`.

For more details, refer to (Spiller et al., 2013).

18.7 Application

18.7.1 `emmeans` Package

The `emmeans` package (Estimated Marginal Means) is a powerful tool for post-hoc analysis of linear models, enabling researchers to explore interaction effects through simple slopes and estimated marginal means.

To install and load the package:

```
install.packages("emmeans")
```

The dataset used in this section is sourced from the UCLA Statistical Consulting Group, where:

- `gender` (male, female) and `prog` (exercise program: jogging, swimming, reading) are categorical variables.
- `loss` represents weight loss, and `hours` and `effort` are continuous predictors.

```
library(tidyverse)
dat <- readRDS("data/exercise.rds") %>%
    mutate(prog = factor(prog, labels = c("jog", "swim", "read"))) %>%
    mutate(gender = factor(gender, labels = c("male", "female")))
```

18.7.1.1 Continuous by Continuous Interaction

We begin with an interaction model between two continuous variables: `hours` (exercise duration) and `effort` (self-reported effort level).

```
contcont <- lm(loss ~ hours * effort, data = dat)
summary(contcont)
#>
#> Call:
#> lm(formula = loss ~ hours * effort, data = dat)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -29.52 -10.60  -1.78  11.13  34.51
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   7.79864   11.60362   0.672   0.5017
#> hours        -9.37568    5.66392  -1.655   0.0982 .
```

```
#> effort       -0.08028    0.38465  -0.209   0.8347
#> hours:effort  0.39335    0.18750   2.098   0.0362 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 13.56 on 896 degrees of freedom
#> Multiple R-squared:  0.07818,    Adjusted R-squared:  0.07509
#> F-statistic: 25.33 on 3 and 896 DF,  p-value: 9.826e-16
```

18.7.1.1.1 Simple Slopes Analysis (Spotlight Analysis) Following Aiken and West (2005), the spotlight analysis examines the effect of `hours` on `loss` at three levels of `effort`:

- Mean of `effort` plus one standard deviation
- Mean of `effort`
- Mean of `effort` minus one standard deviation

```
library(emmeans)
effar <- round(mean(dat$effort) + sd(dat$effort), 1)
effr  <- round(mean(dat$effort), 1)
effbr <- round(mean(dat$effort) - sd(dat$effort), 1)



# Define values for estimation
mylist <- list(effort = c(effbr, effr, effar))

# Compute simple slopes
emtrends(contcont, ~ effort, var = "hours", at = mylist)
#>  effort hours.trend    SE  df lower.CL upper.CL
#>    24.5       0.261 1.352 896   -2.392     2.91
#>    29.7       2.307 0.915 896    0.511     4.10
#>    34.8       4.313 1.308 896    1.745     6.88
#>
#> Confidence level used: 0.95

# Visualization of the interaction
mylist <- list(hours = seq(0, 4, by = 0.4),
               effort = c(effbr, effr, effar))
emmip(contcont, effort ~ hours, at = mylist, CIs = TRUE)
```

```
# Test statistical differences in slopes
emtrends(
    contcont,
    pairwise ~ effort,
    var = "hours",
    at = mylist,
    adjust = "none"
)
#> $emtrends
#>  effort hours.trend    SE  df lower.CL upper.CL
#>    24.5       0.261 1.352 896   -2.392     2.91
#>    29.7       2.307 0.915 896    0.511     4.10
#>    34.8       4.313 1.308 896    1.745     6.88
#>
#> Results are averaged over the levels of: hours
#> Confidence level used: 0.95
#>
#> $contrasts
#>  contrast                estimate    SE  df t.ratio p.value
#>  effort24.5 - effort29.7    -2.05 0.975 896  -2.098  0.0362
#>  effort24.5 - effort34.8    -4.05 1.931 896  -2.098  0.0362
#>  effort29.7 - effort34.8    -2.01 0.956 896  -2.098  0.0362
#>
#> Results are averaged over the levels of: hours
```

The three p-values obtained above correspond to the interaction term in the regression model.

For a professional figure, we refine the visualization using `ggplot2`:

```
library(ggplot2)

# Prepare data for plotting
mylist <- list(hours = seq(0, 4, by = 0.4),
               effort = c(effbr, effr, effar))
contcontdat <-
    emmip(contcont,
          effort ~ hours,
          at = mylist,
          CIs = TRUE,
          plotit = FALSE)

# Convert effort levels to factors
contcontdat$feffort <- factor(contcontdat$effort)
levels(contcontdat$feffort) <- c("low", "medium", "high")

# Generate plot
p  <-
    ggplot(data = contcontdat,
           aes(x = hours, y = yvar, color = feffort)) +
    geom_line()

p1 <-
    p +
    geom_ribbon(aes(ymax = UCL, ymin = LCL, fill = feffort),
                alpha = 0.4)

p1  + labs(x = "Exercise Hours",
           y = "Weight Loss",
           color = "Effort",
           fill = "Effort Level")
```

18.7.1.2 Continuous by Categorical Interaction

Next, we examine an interaction where `hours` (continuous) interacts with `gender` (categorical). We set "Female" as the reference category:

```
dat$gender <- relevel(dat$gender, ref = "female")
contcat <- lm(loss ~ hours * gender, data = dat)
summary(contcat)
#>
#> Call:
#> lm(formula = loss ~ hours * gender, data = dat)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -27.118 -11.350  -1.963  10.001  42.376
#>
#> Coefficients:
#>                  Estimate Std. Error t value Pr(>|t|)
#> (Intercept)         3.335      2.731   1.221    0.222
#> hours               3.315      1.332   2.489    0.013 *
#> gendermale          3.571      3.915   0.912    0.362
#> hours:gendermale   -1.724      1.898  -0.908    0.364
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 14.06 on 896 degrees of freedom
#> Multiple R-squared:  0.008433,   Adjusted R-squared:  0.005113
```

```
#> F-statistic:  2.54 on 3 and 896 DF,  p-value: 0.05523
```

Simple Slopes by Gender

```
# Compute simple slopes for each gender
emtrends(contcat, ~ gender, var = "hours")
#>  gender hours.trend   SE  df lower.CL upper.CL
#>  female        3.32 1.33 896    0.702     5.93
#>  male          1.59 1.35 896   -1.063     4.25
#>
#> Confidence level used: 0.95

# Test slope differences
emtrends(contcat, pairwise ~ gender, var = "hours")
#> $emtrends
#>  gender hours.trend   SE  df lower.CL upper.CL
#>  female        3.32 1.33 896    0.702     5.93
#>  male          1.59 1.35 896   -1.063     4.25
#>
#> Confidence level used: 0.95
#>
#> $contrasts
#>  contrast      estimate  SE  df t.ratio p.value
#>  female - male     1.72 1.9 896   0.908  0.3639
```

Since this test is equivalent to the interaction term in the regression model, a significant result confirms a moderating effect of `gender`.

```
mylist <- list(hours = seq(0, 4, by = 0.4),
               gender = c("female", "male"))
emmip(contcat, gender ~ hours, at = mylist, CIs = TRUE)
```

18.7.1.3 Categorical by Categorical Interaction

Now, we examine the interaction between two categorical variables: `gender` (male, female) and `prog` (exercise program). We set "Read" as the reference category for `prog` and "Female" for `gender`:

```
dat$prog   <- relevel(dat$prog, ref = "read")
dat$gender <- relevel(dat$gender, ref = "female")

catcat <- lm(loss ~ gender * prog, data = dat)
summary(catcat)
#>
#> Call:
#> lm(formula = loss ~ gender * prog, data = dat)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -19.1723  -4.1894  -0.0994   3.7506  27.6939
#>
#> Coefficients:
#>                     Estimate Std. Error t value Pr(>|t|)
#> (Intercept)          -3.6201     0.5322  -6.802 1.89e-11 ***
#> gendermale           -0.3355     0.7527  -0.446    0.656
#> progjog               7.9088     0.7527  10.507  < 2e-16 ***
#> progswim             32.7378     0.7527  43.494  < 2e-16 ***
#> gendermale:progjog    7.8188     1.0645   7.345 4.63e-13 ***
#> gendermale:progswim  -6.2599     1.0645  -5.881 5.77e-09 ***
```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 6.519 on 894 degrees of freedom
#> Multiple R-squared:  0.7875, Adjusted R-squared:  0.7863
#> F-statistic: 662.5 on 5 and 894 DF,  p-value: < 2.2e-16
```

Simple Effects and Contrast Analysis

```
# Estimated marginal means for all combinations of gender and program
emcatcat <- emmeans(catcat, ~ gender * prog)

# Compare effects of gender within each program
contrast(emcatcat, "revpairwise", by = "prog", adjust = "bonferroni")
#> prog = read:
#>  contrast      estimate    SE  df t.ratio p.value
#>  male - female   -0.335 0.753 894  -0.446  0.6559
#>
#> prog = jog:
#>  contrast      estimate    SE  df t.ratio p.value
#>  male - female    7.483 0.753 894   9.942  <.0001
#>
#> prog = swim:
#>  contrast      estimate    SE  df t.ratio p.value
#>  male - female   -6.595 0.753 894  -8.762  <.0001
```

```
emmip(catcat, prog ~ gender, CIs = TRUE)
```

For a more intuitive presentation, we use a bar graph with error bars

```
# Prepare data
catcatdat <- emmip(catcat,
                   gender ~ prog,
                   CIs = TRUE,
                   plotit = FALSE)

# Generate plot
p <-
    ggplot(data = catcatdat,
           aes(x = prog, y = yvar, fill = gender)) +
    geom_bar(stat = "identity", position = "dodge")

p1 <-
    p + geom_errorbar(
        position = position_dodge(.9),
        width = .25,
        aes(ymax = UCL, ymin = LCL),
        alpha = 0.3
    )

p1  + labs(x = "Exercise Program",
           y = "Weight Loss",
           fill = "Gender")
```

18.7.2 `probemod` Package

The `probemod` package is designed for moderation analysis, particularly focusing on **Johnson-Neyman intervals** and **simple slopes analysis**. However, **this package is not recommended** due to known issues with subscript handling and formatting errors in some outputs.

```
install.packages("probemod", dependencies = T)
```

The **Johnson-Neyman technique** identifies values of the moderator (`gender`) where the effect of the independent variable (`hours`) on the dependent variable (`loss`) is statistically significant. This method is particularly useful when the moderator is continuous but can also be applied to categorical moderators.

Example: J-N Analysis in a `loss ~ hours * gender` Model

```
library(probemod)

myModel <-
    lm(loss ~ hours * gender, data = dat %>%
           select(loss, hours, gender))

jnresults <- jn(myModel,
                dv = 'loss',
                iv = 'hours',
                mod = 'gender')
```

The `jn()` function computes Johnson-Neyman intervals, highlighting the values of `gender` at which the relationship between `hours` and `loss` is statistically significant.

The **Pick-a-Point** method tests the simple effect of `hours` at specific values of `gender`, akin to spotlight analysis.

```
pickapoint(
    myModel,
    dv = 'loss',
    iv = 'hours',
    mod = 'gender',
    alpha = .01
)

plot(jnresults)
```

18.7.3 `interactions` Package

The `interactions` package is a **recommended** tool for visualizing and interpreting interaction effects in regression models. It provides user-friendly functions for **interaction plots**, **simple slopes analysis**, and **Johnson-Neyman**

intervals, making it an excellent choice for moderation analysis.

```
install.packages("interactions")
```

18.7.3.1 Continuous by Continuous Interaction

This section covers interactions where at least one of the two variables is continuous.

Example: Interaction Between `Illiteracy` and `Murder`

We use the `state.x77` dataset to explore how **Illiteracy Rate** and **Murder Rate** interact to predict **Income** across U.S. states.

```
states <- as.data.frame(state.x77)
fiti <- lm(Income ~ Illiteracy * Murder + `HS Grad`, data = states)
summary(fiti)
#>
#> Call:
#> lm(formula = Income ~ Illiteracy * Murder + `HS Grad`, data = states)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -916.27 -244.42   28.42  228.14 1221.16
#>
#> Coefficients:
#>                   Estimate Std. Error t value Pr(>|t|)
#> (Intercept)        1414.46     737.84   1.917  0.06160 .
#> Illiteracy          753.07     385.90   1.951  0.05724 .
#> Murder              130.60      44.67   2.923  0.00540 **
#> `HS Grad`            40.76      10.92   3.733  0.00053 ***
#> Illiteracy:Murder   -97.04      35.86  -2.706  0.00958 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 459.5 on 45 degrees of freedom
#> Multiple R-squared:  0.4864, Adjusted R-squared:  0.4407
#> F-statistic: 10.65 on 4 and 45 DF,  p-value: 3.689e-06
```

For continuous moderators, the standard values chosen for visualization are:

- **Mean + 1 SD**
- **Mean**
- **Mean - 1 SD**

The `interact_plot()` function provides an easy way to visualize these effects.

```r
library(interactions)
interact_plot(fiti,
              pred = Illiteracy,
              modx = Murder,

              # Disable automatic mean-centering
              centered = "none",

              # Exclude the mean value of the moderator
              # modx.values = "plus-minus",

              # Divide the moderator's distribution into three groups
              # modx.values = "terciles",

              plot.points = TRUE, # Overlay raw data

              # Different shapes for different levels of the moderator
              point.shape = TRUE,

              # Jittering to prevent overplotting
              jitter = 0.1,

              # Custom appearance
              x.label = "Illiteracy Rate (%)",
              y.label = "Income ($)",
              main.title = "Interaction Between Illiteracy and Murder Rate",
              legend.main = "Murder Rate Levels",
              colors = "blue",

              # Confidence bands
              interval = TRUE,
              int.width = 0.9,
              robust = TRUE # Use robust standard errors
              )
```

If the model includes **weights**, they can be incorporated into the visualization

```
fiti <- lm(Income ~ Illiteracy * Murder,
           data = states,
           weights = Population)

interact_plot(fiti,
              pred = Illiteracy,
              modx = Murder,
              plot.points = TRUE)
```

A **partial effect plot** shows how the effect of one variable changes across different levels of another variable while controlling for other predictors.

```
library(ggplot2)
data(cars)

fitc <- lm(cty ~ year + cyl * displ + class + fl + drv,
           data = mpg)
summary(fitc)
#>
#> Call:
#> lm(formula = cty ~ year + cyl * displ + class + fl + drv, data = mpg)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -5.9772 -0.7164  0.0018  0.7690  5.9314
#>
#> Coefficients:
#>                   Estimate Std. Error t value Pr(>|t|)
#> (Intercept)     -200.97599   47.00954  -4.275 2.86e-05 ***
#> year               0.11813    0.02347   5.033 1.01e-06 ***
#> cyl               -1.85648    0.27745  -6.691 1.86e-10 ***
#> displ             -3.56467    0.65943  -5.406 1.70e-07 ***
#> classcompact      -2.60177    0.92972  -2.798 0.005597 **
#> classmidsize      -2.62996    0.93273  -2.820 0.005253 **
#> classminivan      -4.40817    1.03844  -4.245 3.24e-05 ***
#> classpickup       -4.37322    0.93416  -4.681 5.02e-06 ***
```

```
#> classsubcompact   -2.38384    0.92943  -2.565 0.010997 *
#> classsuv          -4.27352    0.86777  -4.925 1.67e-06 ***
#> fld                6.34343    1.69499   3.742 0.000233 ***
#> fle               -4.57060    1.65992  -2.754 0.006396 **
#> flp               -1.91733    1.58649  -1.209 0.228158
#> flr               -0.78873    1.56551  -0.504 0.614901
#> drvf               1.39617    0.39660   3.520 0.000525 ***
#> drvr               0.48740    0.46113   1.057 0.291694
#> cyl:displ          0.36206    0.07934   4.564 8.42e-06 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.526 on 217 degrees of freedom
#> Multiple R-squared:  0.8803, Adjusted R-squared:  0.8715
#> F-statistic: 99.73 on 16 and 217 DF,  p-value: < 2.2e-16

interact_plot(
    fitc,
    pred = displ,
    modx = cyl,

    # Show observed data as partial residuals
    partial.residuals = TRUE,

    # Specify moderator values manually
    modx.values = c(4, 5, 6, 8)
)
```

To check whether an interaction is **truly linear**, we can compare fitted lines based on:

- **The whole sample** (black line)
- **Subsamples** based on the moderator (red line)

```
# Generate synthetic data
x_2 <- runif(n = 200, min = -3, max = 3)
w   <- rbinom(n = 200, size = 1, prob = 0.5)
err <- rnorm(n = 200, mean = 0, sd = 4)
y_2 <- 2.5 - x_2 ^ 2 - 5 * w + 2 * w * (x_2 ^ 2) + err

data_2 <- as.data.frame(cbind(x_2, y_2, w))

# Fit interaction model
model_2 <- lm(y_2 ~ x_2 * w, data = data_2)
summary(model_2)
#>
#> Call:
#> lm(formula = y_2 ~ x_2 * w, data = data_2)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -13.7333  -3.4236   0.2435   3.6085  10.3728
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
```

```
#> (Intercept)  -0.7082     0.5076  -1.395   0.1645
#> x_2          -0.2600     0.3058  -0.850   0.3962
#> w             1.6177     0.7054   2.293   0.0229 *
#> x_2:w         0.3871     0.4116   0.940   0.3481
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 4.972 on 196 degrees of freedom
#> Multiple R-squared:  0.03222,    Adjusted R-squared:  0.01741
#> F-statistic: 2.175 on 3 and 196 DF,  p-value: 0.09223

# Linearity check plot
interact_plot(
    model_2,
    pred = x_2,
    modx = w,
    linearity.check = TRUE,
    plot.points = TRUE
)
```

18.7.3.1.1 Simple Slopes Analysis A **simple slopes analysis** examines the **conditional effect** of an independent variable (X) at specific levels of the moderator (M).

How `sim_slopes()` Works:

- **Continuous moderators**: Analyzes effects at the **mean** and **±1 SD**.

- **Categorical moderators**: Uses all factor levels.

- **Mean-centers all variables** except the predictor of interest.

Example: Continuous by Continuous Interaction

```
sim_slopes(fiti,
           pred = Illiteracy,
           modx = Murder,
           johnson_neyman = FALSE)
#> SIMPLE SLOPES ANALYSIS
#>
#> Slope of Illiteracy when Murder =  5.420973 (- 1 SD):
#>
#>     Est.     S.E.   t val.      p
#> -------- -------- -------- ------
#>   -71.59   268.65    -0.27   0.79
#>
#> Slope of Illiteracy when Murder =  8.685043 (Mean):
#>
#>      Est.     S.E.   t val.      p
#> --------- -------- -------- ------
#>   -437.12   175.82    -2.49   0.02
#>
#> Slope of Illiteracy when Murder = 11.949113 (+ 1 SD):
#>
#>      Est.     S.E.   t val.      p
#> --------- -------- -------- ------
#>   -802.66   145.72    -5.51   0.00
```

We can also visualize the simple slopes

```
# Store results
ss <- sim_slopes(fiti,
                 pred = Illiteracy,
                 modx = Murder,
                 modx.values = c(0, 5, 10))

# Plot the slopes
plot(ss)
```

For publication-quality results, we convert the simple slopes analysis into a table using `huxtable`.

```
library(huxtable)

ss <- sim_slopes(fiti,
                 pred = Illiteracy,
                 modx = Murder,
                 modx.values = c(0, 5, 10))

# Convert to a formatted table
as_huxtable(ss)
```

Table 18.1

Value of Murder	Slope of Illiteracy
Value of Murder	slope
0.00	535.50 (458.77)
5.00	-24.44 (282.48)
10.00	-584.38 (152.37)***

18.7.3.1.2 Johnson-Neyman Intervals The **Johnson-Neyman technique** identifies the range of the moderator (M) where the effect of the

predictor (X) on the dependent variable (Y) is statistically significant. This approach is useful when the moderator is continuous, allowing us to determine where an effect exists rather than arbitrarily choosing values.

Although the J-N approach has been widely used (Johnson and Neyman, 1936), it has known **inflated Type I error rates** (Bauer and Curran, 2005). A correction method was proposed by (Esarey and Sumner, 2018) to address these issues.

Since J-N performs multiple comparisons across all values of the moderator, it **inflates Type I error**. To control for this, we use False Discovery Rate correction.

Example: Johnson-Neyman Analysis

```
sim_slopes(
    fiti,
    pred = Illiteracy,
    modx = Murder,
    johnson_neyman = TRUE,
    control.fdr = TRUE,  # Correction for Type I and II errors

    # Include conditional intercepts
    # cond.int = TRUE,

    robust = "HC3",  # Use robust SE

    # Don't mean-center non-focal variables
    # centered = "none",

    jnalpha = 0.05  # Significance level
)
#> JOHNSON-NEYMAN INTERVAL
#>
#> When Murder is OUTSIDE the interval [-11.70, 8.75], the slope of Illiteracy
#> is p < .05.
#>
#> Note: The range of observed values of Murder is [1.40, 15.10]
#>
#> Interval calculated using false discovery rate adjusted t = 2.33
#>
#> SIMPLE SLOPES ANALYSIS
#>
#> Slope of Illiteracy when Murder =  5.420973 (- 1 SD):
#>
#>     Est.     S.E.   t val.      p
#> -------- -------- -------- ------
```

```
#>   -71.59   256.60    -0.28   0.78
#>
#> Slope of Illiteracy when Murder =  8.685043 (Mean):
#>
#>      Est.     S.E.   t val.      p
#> --------- -------- -------- ------
#>   -437.12   191.07    -2.29   0.03
#>
#> Slope of Illiteracy when Murder = 11.949113 (+ 1 SD):
#>
#>      Est.     S.E.   t val.      p
#> --------- -------- -------- ------
#>   -802.66   178.75    -4.49   0.00
```

To visualize the J-N intervals

```
johnson_neyman(fiti,
               pred = Illiteracy,
               modx = Murder,
               control.fdr = TRUE, # Corrects for Type I error
               alpha = .05)
#> JOHNSON-NEYMAN INTERVAL
#>
#> When Murder is OUTSIDE the interval [-22.57, 8.52], the slope of Illiteracy
#> is p < .05.
#>
#> Note: The range of observed values of Murder is [1.40, 15.10]
#>
#> Interval calculated using false discovery rate adjusted t = 2.33
```

- The y-axis represents the conditional slope of the predictor (X).
- The x-axis represents the values of the moderator (M).
- The shaded region represents the range where the effect of X on Y is statistically significant.

18.7.3.1.3 Three-Way Interactions In three-way interactions, the effect of X on Y depends on **two moderators** (M_1 and M_2). This allows for a more nuanced understanding of moderation effects.

Example: 3-Way Interaction Visualization

```
library(jtools)
# Convert 'cyl' to factor
mtcars$cyl <- factor(mtcars$cyl,
                     labels = c("4 cylinder", "6 cylinder", "8 cylinder"))

# Fit the model
fitc3 <- lm(mpg ~ hp * wt * cyl, data = mtcars)

# Plot interaction
interact_plot(fitc3,
              pred = hp,
              modx = wt,
              mod2 = cyl) +
    theme_apa(legend.pos = "bottomright")
```

18.7.3.1.4 Johnson-Neyman for Three-Way Interaction The Johnson-Neyman technique can also be applied in a **three-way interaction context**

```
library(survey)
data(api)

# Define survey design
dstrat <- svydesign(
    id = ~ 1,
    strata = ~ stype,
    weights = ~ pw,
    data = apistrat,
    fpc = ~ fpc
)

# Fit 3-way interaction model
regmodel3 <-
    survey::svyglm(api00 ~ avg.ed * growth * enroll, design = dstrat)

# Johnson-Neyman analysis with visualization
sim_slopes(
    regmodel3,
    pred = growth,
    modx = avg.ed,
    mod2 = enroll,
    jnplot = TRUE
)
```

```
#> ############### While enroll (2nd moderator) =  153.0518 (- 1 SD) ##############
#>
#> JOHNSON-NEYMAN INTERVAL
#>
#> When avg.ed is OUTSIDE the interval [2.75, 3.82], the slope of growth is p
#> < .05.
#>
#> Note: The range of observed values of avg.ed is [1.38, 4.44]
#>
#> SIMPLE SLOPES ANALYSIS
#>
#> Slope of growth when avg.ed = 2.085002 (- 1 SD):
#>
#>   Est.   S.E.   t val.      p
#> ------ ------ -------- ------
#>   1.25   0.32     3.86   0.00
#>
#> Slope of growth when avg.ed = 2.787381 (Mean):
#>
#>   Est.   S.E.   t val.      p
#> ------ ------ -------- ------
#>   0.39   0.22     1.75   0.08
#>
#> Slope of growth when avg.ed = 3.489761 (+ 1 SD):
#>
#>    Est.   S.E.   t val.      p
#> ------- ------ -------- ------
#>   -0.48   0.35    -1.37   0.17
#>
#> ################ While enroll (2nd moderator) =  595.2821 (Mean) ###############
#>
#> JOHNSON-NEYMAN INTERVAL
#>
#> When avg.ed is OUTSIDE the interval [2.84, 7.83], the slope of growth is p
#> < .05.
#>
#> Note: The range of observed values of avg.ed is [1.38, 4.44]
#>
#> SIMPLE SLOPES ANALYSIS
#>
#> Slope of growth when avg.ed = 2.085002 (- 1 SD):
#>
#>   Est.   S.E.   t val.      p
#> ------ ------ -------- ------
#>   0.72   0.22     3.29   0.00
```

```
#>
#> Slope of growth when avg.ed = 2.787381 (Mean):
#>
#>   Est.   S.E.   t val.      p
#> ------ ------ -------- ------
#>   0.34   0.16     2.16   0.03
#>
#> Slope of growth when avg.ed = 3.489761 (+ 1 SD):
#>
#>    Est.   S.E.   t val.      p
#> ------- ------ -------- ------
#>   -0.04   0.24    -0.16   0.87
#>
#> ############### While enroll (2nd moderator) = 1037.5125 (+ 1 SD) ##############
#>
#> JOHNSON-NEYMAN INTERVAL
#>
#> The Johnson-Neyman interval could not be found. Is the p value for your
#> interaction term below the specified alpha?
#>
#> SIMPLE SLOPES ANALYSIS
#>
#> Slope of growth when avg.ed = 2.085002 (- 1 SD):
#>
#>   Est.   S.E.   t val.      p
#> ------ ------ -------- ------
#>   0.18   0.31     0.58   0.56
#>
#> Slope of growth when avg.ed = 2.787381 (Mean):
#>
#>   Est.   S.E.   t val.      p
#> ------ ------ -------- ------
#>   0.29   0.20     1.49   0.14
#>
#> Slope of growth when avg.ed = 3.489761 (+ 1 SD):
#>
#>   Est.   S.E.   t val.      p
#> ------ ------ -------- ------
#>   0.40   0.27     1.49   0.14
```

To present the results in a **publication-ready format**, we generate tables and plots

```
ss3 <-
    sim_slopes(regmodel3,
               pred = growth,
               modx = avg.ed,
               mod2 = enroll)

# Plot results
plot(ss3)
```

```
# Convert results into a formatted table
library(huxtable)
as_huxtable(ss3)
```

18.7.3.2 Categorical Interactions

Interactions between categorical predictors can be visualized using **categorical plots**.

Example: Interaction Between `cyl`, `fwd`, and `auto`

```
library(ggplot2)

# Convert variables to factors
mpg2 <- mpg %>%
    mutate(cyl = factor(cyl))

mpg2["auto"] <- "auto"
mpg2$auto[mpg2$trans %in% c("manual(m5)", "manual(m6)")] <- "manual"
mpg2$auto <- factor(mpg2$auto)

mpg2["fwd"] <- "2wd"
mpg2$fwd[mpg2$drv == "4"] <- "4wd"
mpg2$fwd <- factor(mpg2$fwd)

# Drop cars with 5 cylinders (since most have 4, 6, or 8)
mpg2 <- mpg2[mpg2$cyl != "5",]
```

Table 18.2

enroll = 153	
Value of avg.ed	Slope of growth
Value of avg.ed	slope
2.09	1.25 (0.32)***
2.79	0.39 (0.22)#
enroll = 595.28	
Value of avg.ed	Slope of growth
3.49	-0.48 (0.35)
2.09	0.72 (0.22)**
2.79	0.34 (0.16)*
enroll = 1037.51	
Value of avg.ed	Slope of growth
3.49	-0.04 (0.24)
2.09	0.18 (0.31)
2.79	0.29 (0.20)
3.49	0.40 (0.27)

```r
# Fit the model
fit3 <- lm(cty ~ cyl * fwd * auto, data = mpg2)

library(jtools) # For summ()
summ(fit3)
```

Observations	230
Dependent variable	cty
Type	OLS linear regression

```r
cat_plot(fit3,
         pred = cyl,
```

F(11,218)	61.37
R^2	0.76
Adj. R^2	0.74

	Est.	S.E.	t val.	p
(Intercept)	21.37	0.39	54.19	0.00
cyl6	-4.37	0.54	-8.07	0.00
cyl8	-8.37	0.67	-12.51	0.00
fwd4wd	-2.91	0.76	-3.83	0.00
automanual	1.45	0.57	2.56	0.01
cyl6:fwd4wd	0.59	0.96	0.62	0.54
cyl8:fwd4wd	2.13	0.99	2.15	0.03
cyl6:automanual	-0.76	0.90	-0.84	0.40
cyl8:automanual	0.71	1.18	0.60	0.55
fwd4wd:automanual	-1.66	1.07	-1.56	0.12
cyl6:fwd4wd:automanual	1.29	1.52	0.85	0.40
cyl8:fwd4wd:automanual	-1.39	1.76	-0.79	0.43

Standard errors: OLS

```
        modx = fwd,
        plot.points = TRUE)
```

Line Plot for Categorical Interaction

```
cat_plot(
    fit3,
    pred = cyl,
    modx = fwd,
    geom = "line",
    point.shape = TRUE,
    vary.lty = TRUE
)
```

Bar Plot Representation

```
cat_plot(
    fit3,
    pred = cyl,
    modx = fwd,
    geom = "bar",
    interval = TRUE,
    plot.points = TRUE
)
```

18.7.4 `interactionR` Package

The `interactionR` package is designed for publication-quality reporting of interaction effects, particularly in epidemiology and social sciences. It provides tools for computing interaction measures, confidence intervals, and statistical inference following well-established methodologies.

Key Features:

- Publication-Ready Interaction Analysis
- Confidence intervals calculated using:
  - Delta method (Hosmer and Lemeshow, 1992)
  - Variance recovery ("mover") method (Zou, 2008)
  - Bootstrapping (Assmann et al., 1996)
- Standardized reporting guidelines based on (Knol and VanderWeele, 2012).

```
install.packages("interactionR", dependencies = T)
```

18.7.5 `sjPlot` Package

The `sjPlot` package is highly recommended for publication-quality visualizations of interaction effects. It provides enhanced aesthetics and customizable interaction plots suitable for academic journals.

More details: sjPlot interaction visualization

```
install.packages("sjPlot")
```

18.7.6 Summary of Moderation Analysis Packages

Package	Purpose	Key Features	Recommended?
`emmeans`	Estimated marginal means & simple slopes	Computes simple slopes, spotlight analysis, floodlight analysis (J-N method)	Yes
`probemod`	Johnson-Neyman technique	Tests moderator significance ranges	No (Subscript issues)
`interactions`	Interaction visualization	Produces robust, customizable interaction plots	Yes
`interactionR`	Epidemiological interaction measures	Computes RERI, AP, SI for additive scale interactions	Yes (for public health research)
`sjPlot`	Publication-quality interaction plots	Highly customizable, ideal for academic papers	Highly Recommended

Chapter 19

Mediation

19.1 Traditional Approach

The classical mediation analysis follows the approach introduced by Baron and Kenny (1986), though it has limitations, particularly in requiring the first step ($X \to Y$) to be significant. Despite its shortcomings, this framework provides a useful foundation.

19.1.1 Steps in the Traditional Mediation Model

Mediation is typically assessed through three regression models:

1. **Total Effect**: $X \to Y$
2. **Path** a: $X \to M$
3. **Path** b and Direct Effect (c'): $X + M \to Y$

where:

- X = independent (causal) variable
- Y = dependent (outcome) variable
- M = mediating variable

Originally, Baron and Kenny (1986) required the direct path $X \to Y$ to be significant. However, mediation can still occur even if this direct effect is not significant. For example:

- The effect of X on Y might be **fully absorbed** by M.
- Multiple mediators (M_1, M_2) with **opposing effects** could cancel each other out, leading to a non-significant direct effect.

19.1.2 Graphical Representation of Mediation

19.1.2.1 Unmediated Model

Figure 19.1: Unmediated model

Here, c represents the **total effect** of X on Y.

19.1.2.2 Mediated Model

Here:

- $c' =$ **direct effect** (effect of X on Y after accounting for mediation)
- $ab =$ **indirect effect** (mediation pathway)

Thus, we can express:

$$\text{total effect} = \text{direct effect} + \text{indirect effect}$$

or,

$$c = c' + ab$$

This equation holds under standard linear models but not necessarily in cases such as:

1. Latent variable models
2. Logistic regression (only an approximation)
3. Multilevel models (Bauer et al., 2006)

19.1.3 Measuring Mediation

Several approaches exist for quantifying the **indirect effect** (ab):

1. **Proportional Reduction Approach**:

$$1 - \frac{c'}{c}$$

   Not recommended due to high instability, especially when c is small (MacKinnon et al., 1995).

2. **Product Method**:

$$a \times b$$

   The most common approach.

3. **Difference Method**:

$$c - c'$$

   Conceptually similar to the product method but less precise in small samples.

19.1.4 Assumptions in Linear Mediation Models

For valid mediation analysis, the following assumptions should hold:

1. **No unmeasured confounders** between $X - Y$, $X - M$, and $M - Y$.
2. **No reverse causality**: X should not be influenced by a confounder (C) that also affects $M - Y$.
3. **Measurement reliability**: M should be measured without error (if not, consider errors-in-variables models).

Regression Equations for Mediation Steps

Step 1: Total Effect of X on Y

$$Y = \beta_0 + cX + \epsilon$$

- The significance of c is **not required** for mediation to occur.

Step 2: Effect of X on M

$$M = \alpha_0 + aX + \epsilon$$

- The coefficient a must be significant for mediation analysis to proceed.

Step 3: Effect of M on Y (Including X)

$$Y = \gamma_0 + c'X + bM + \epsilon$$

- If c' becomes non-significant after including M, full mediation occurs.
- If c' is reduced but remains significant, partial mediation is present.

Table 19.1: Interpretation of Mediation Outcomes

Effect of X on Y	Mediation Type
b_4 (from Step 3) is **insignificant**	Full mediation
$b_4 < b_1$ (from Step 1) but still significant	Partial mediation

---

19.1.5 Testing for Mediation

Several statistical tests exist to assess whether the indirect effect (ab) is significant:

1. **Sobel Test** (Sobel, 1982)
   - Based on the standard error of ab.
   - **Limitation**: Assumes normality of ab, which may not hold in small samples.
2. **Joint Significance Test**
   - If both a and b are significant, mediation is likely.
3. **Bootstrapping (Preferred)** (Preacher and Hayes, 2004, Shrout and Bolger (2002))
   - Estimates the confidence interval for ab.
   - Does not assume normality.
   - Recommended for small-to-moderate sample sizes.

19.1.6 Additional Considerations

- Proximal mediation (where path a exceeds path b) can lead to multicollinearity and reduced statistical power. In contrast, distal mediation (where path b exceeds path a) tends to maximize power. In fact, slightly distal mediators—where b is somewhat larger than a—often strike an ideal balance for power in mediation analyses (Hoyle, 1999).
- Tests of direct effects (c and c') generally have lower power than tests of the indirect effect (ab). As a result, it is possible for the indirect effect (ab) to be statistically significant even when the direct effect (c) is not. This situation can appear to indicate "complete mediation," yet the lack of a statistically significant direct effect between X and Y (i.e., c') does not definitively rule out other possibilities (Kenny and Judd, 2014).
- Because testing ab essentially combines two tests, it often provides a power advantage over testing c' alone. However, using a non-significant c' as the sole criterion for claiming complete mediation should be done cautiously—if at all—given the importance of adequate sample size and power. Indeed, Hayes and Scharkow (2013) recommend avoiding claims of complete mediation based solely on a non-significant c', particularly when partial mediation may still be present.

19.1.7 Assumptions in Mediation Analysis

Valid mediation analysis requires several key assumptions, which can be categorized into **causal direction**, **interaction effects**, **measurement reliability**, and **confounding control**.

19.1.7.1 Direction

- Causal Order of Variables
- A simple but weak solution is to measure X before M and Y to prevent reverse causality (i.e., M or Y causing X). Similarly, measuring M before Y avoids feedback effects of Y on M.
- However, causal feedback loops between M and Y may still exist.
  - If we assume full mediation ($c' = 0$), models with reciprocal causal effects between M and Y can be estimated using instrumental variables (IV).
  - Smith (1982) suggests treating both M and Y as potential mediators of each other, requiring distinct instrumental variables for each to avoid cross-contamination of causal effects.

19.1.7.2 Interaction Effects in Mediation

- If M interacts with X in predicting Y, then M is both a mediator and a moderator (Baron and Kenny, 1986).
- **The interaction term** $X \times M$ should always be included in the model to account for possible moderation effects.
- For interpreting such interactions in mediation models, see (VanderWeele, 2015), who provides a framework for **moderated mediation** analysis.

19.1.7.3 Reliability

Measurement error in any of the three key variables (X, M, Y) can bias estimates of mediation effects.

1. Measurement Error in the Mediator (M):
   - Biases both b and c'.
   - Potential solution: Model M as a latent variable (reduces bias but may decrease statistical power) (Ledgerwood and Shrout, 2011).
   - Specific effects:
     - b is attenuated (biased toward 0).
     - c' is:
       - Overestimated if $ab > 0$.
       - Underestimated if $ab < 0$.
2. Measurement Error in the Treatment (X):
   - Biases both a and b.
   - Specific effects:
     - a is attenuated.
     - b is:
       - Overestimated if $ac' > 0$.
       - Underestimated if $ac' < 0$.
3. Measurement Error in the Outcome (Y):
   - If unstandardized, there is no bias.
   - If standardized, there is attenuation bias (reduced effect sizes due to error variance).

19.1.7.4 Confounding in Mediation Analysis

Omitted variable bias can distort any of the three core relationships ($X \to Y$, $X \to M$, $M \to Y$). Addressing confounding requires either design-based or statistical solutions.

Design-Based Strategies (Preferred if Feasible)

- Randomization of the independent variable (X) reduces confounding bias.

- Randomization of the mediator (M), if possible, further strengthens causal claims.
- Controlling for measured confounders, though this only addresses observable confounding.

Statistical Strategies (When Randomization is Not Possible)

1. Instrumental Variables Approach:
   - Used when a confounder affects both M and Y.
   - Front-door adjustment can be applied if there exists a third variable that fully mediates the effect of M on Y while being independent of the confounder.
2. Weighting Methods (e.g., Inverse Probability Weighting - IPW):
   - Corrects for confounding by reweighting observations to balance confounders across treatment groups.
   - Requires the strong ignorability assumption: All confounders must be measured and correctly specified (Westfall and Yarkoni, 2016).
   - While this assumption cannot be formally tested, sensitivity analyses can help assess robustness.
   - See Heiss for R code on implementing IPW in mediation models.

19.1.8 Indirect Effect Tests

Testing the indirect effect (ab) is crucial in mediation analysis. Several methods exist, each with its advantages and limitations.

19.1.8.1 Sobel Test (Delta Method)

- Developed by Sobel (1982).
- Also known as the delta method.
- Not recommended because it assumes the sampling distribution of ab is normal, which often does not hold (MacKinnon et al., 1995).

The standard error (SE) of the indirect effect is:

$$SE_{ab} = \sqrt{\hat{b}^2 s_{\hat{a}}^2 + \hat{a}^2 s_{\hat{b}}^2}$$

The **Z-statistic** for testing whether ab is significantly different from 0 is:

$$z = \frac{\hat{ab}}{\sqrt{\hat{b}^2 s_{\hat{a}}^2 + \hat{a}^2 s_{\hat{b}}^2}}$$

Disadvantages

- Assumes a and b are independent.
- Assumes ab follows a normal distribution.
- Poor performance in small samples.
- Lower power and more conservative than bootstrapping.

Special Case: Inconsistent Mediation

- Mediation can occur even when **direct and indirect effects** have **opposite signs**, known as **inconsistent mediation** (MacKinnon et al., 2007).
- This happens when the **mediator acts as a suppressor variable**, leading to counteracting paths.

```
library(bda)
library(mediation)
data("boundsdata")

# Sobel Test for Mediation
bda::mediation.test(boundsdata$med, boundsdata$ttt, boundsdata$out) |>
    tibble::rownames_to_column() |>
    causalverse::nice_tab(2)
#>   rowname Sobel Aroian Goodman
#> 1 z.value  4.05   4.03    4.07
#> 2 p.value  0.00   0.00    0.00
```

19.1.8.2 Joint Significance Test

- Tests if the indirect effect is nonzero by checking whether both a and b are statistically significant.
- Assumes $a \perp b$ (independence of paths).
- Performs similarly to bootstrapping (Hayes and Scharkow, 2013).
- More robust to non-normality but can be sensitive to heteroscedasticity (Fossum and Montoya, 2023).
- Does not provide confidence intervals, making effect size interpretation harder.

19.1.8.3 Bootstrapping (Preferred Method)

- First applied to mediation by Bollen and Stine (1990).
- Uses resampling to empirically estimate the sampling distribution of the indirect effect.
- Does not require normality assumptions or $a \perp b$ independence.
- Works well with small samples.
- Can handle complex models.

Which Bootstrapping Method?

- **Percentile bootstrap** is preferred due to better **Type I error rates** (Tibbe and Montoya, 2022).
- **Bias-corrected bootstrapping** can be too **liberal** (inflates Type I errors) (Fritz et al., 2012).

Special Case: Meta-Analytic Bootstrapping

- Bootstrapping can be applied **without raw data**, using only $a, b, var(a), var(b), cov(a, b)$ from multiple studies.

```
# Meta-Analytic Bootstrapping for Mediation
library(causalverse)

result <- causalverse::med_ind(
    a = 0.5,
    b = 0.7,
    var_a = 0.04,
    var_b = 0.05,
    cov_ab = 0.01
)
result$plot
```

When an **instrumental variable (IV)** is available, the causal effect can be estimated more reliably. Below are visual representations.

```
library(DiagrammeR)

# Simple Treatment-Outcome Model
grViz("
```

```
digraph {
  graph []
  node [shape = plaintext]
    X [label = 'Treatment']
    Y [label = 'Outcome']
  edge [minlen = 2]
    X->Y
  { rank = same; X; Y }
}")

# Mediation Model with an Instrument
grViz("
digraph {
  graph []
  node [shape = plaintext]
    X [label ='Treatment', shape = box]
    Y [label ='Outcome', shape = box]
    M [label ='Mediator', shape = box]
    IV [label ='Instrument', shape = box]
  edge [minlen = 2]
    IV->X
    X->M
    M->Y
    X->Y
  { rank = same; X; Y; M }
}")
```

Mediation Analysis with Fixed Effects Models

```
library(mediation)
library(fixest)

data("boundsdata")

# Step 1: Total Effect (c)
out1 <- feols(out ~ ttt, data = boundsdata)

# Step 2: Indirect Effect (a)
out2 <- feols(med ~ ttt, data = boundsdata)

# Step 3: Direct & Indirect Effect (c' & b)
out3 <- feols(out ~ med + ttt, data = boundsdata)

# Proportion of Mediation
coef(out2)['ttt'] * coef(out3)['med'] / coef(out1)['ttt'] * 100
#>      ttt
```

```
#> 68.63609
```

Bootstrapped Mediation Analysis

```
library(boot)
set.seed(1)

# Define the bootstrapping function
mediation_fn <- function(data, i) {
    df <- data[i,]

    a_path <- feols(med ~ ttt, data = df)
    a <- coef(a_path)['ttt']

    b_path <- feols(out ~ med + ttt, data = df)
    b <- coef(b_path)['med']

    cp <- coef(b_path)['ttt']

    # Indirect Effect (a * b)
    ind_ef <- a * b
    total_ef <- a * b + cp
    return(c(ind_ef, total_ef))
}

# Perform Bootstrapping
boot_med <- boot(boundsdata, mediation_fn, R = 100, parallel = "multicore", ncpus = 2)
boot_med
#>
#> ORDINARY NONPARAMETRIC BOOTSTRAP
#>
#>
#> Call:
#> boot(data = boundsdata, statistic = mediation_fn, R = 100, parallel = "multicore",
#>     ncpus = 2)
#>
#>
#> Bootstrap Statistics :
#>       original        bias    std. error
#> t1* 0.04112035  0.0006346725 0.009539903
#> t2* 0.05991068 -0.0004462572 0.029556611

# Summary and Confidence Intervals
summary(boot_med) |> causalverse::nice_tab()
#>     R original bootBias bootSE bootMed
#> 1 100     0.04        0   0.01    0.04
```

```
#> 2 100     0.06        0   0.03    0.06

# Confidence Intervals (percentile bootstrap preferred)
boot.ci(boot_med, type = c("norm", "perc"))
#> BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
#> Based on 100 bootstrap replicates
#>
#> CALL :
#> boot.ci(boot.out = boot_med, type = c("norm", "perc"))
#>
#> Intervals :
#> Level      Normal              Percentile
#> 95%   ( 0.0218,  0.0592 )   ( 0.0249,  0.0623 )
#> Calculations and Intervals on Original Scale
#> Some percentile intervals may be unstable

# Point Estimates (Indirect and Total Effects)
colMeans(boot_med$t)
#> [1] 0.04175502 0.05946442
```

Alternatively, use the `robmed` package for **robust** mediation analysis:

```
library(robmed)
```

19.1.9 Power Analysis for Mediation

To assess whether the study has sufficient **power** to detect mediation effects, use:

```
library(pwr2ppl)

# Power analysis for the indirect effect (ab path)
medjs(
    rx1m1 = .3,  # Correlation: X → M (path a)
    rx1y  = .1,  # Correlation: X → Y (path c')
    rym1  = .3,  # Correlation: M → Y (path b)
    n     = 100, # Sample size
    alpha = 0.05,
    mvars = 1,   # Number of mediators
    rep   = 1000 # Replications (use 10,000 for accuracy)
)
```

For **interactive** power analysis, see Kenny's Mediation Power App.

Summary of Indirect Effect Tests

Test	Pros	Cons
Sobel Test	Simple, fast	Assumes normality, low power
Joint Significance Test	Robust to non-normality	No confidence interval
Bootstrapping (Recommended)	No normality assumption, handles small samples	May be liberal if bias-corrected

19.1.10 Multiple Mediation Analysis

In some cases, a single mediator (M) does not fully capture the indirect effect of X on Y. Multiple mediation models extend traditional mediation by including two or more mediators, allowing us to examine how multiple pathways contribute to an outcome.

Several R packages handle multiple mediation models:

- **manymome**: A flexible package for multiple mediation modeling.
  - Vignette

```
library(manymome)
```

- **mma**: Used for multiple mediator models.
  - Package PDF
  - Vignette

```
library(mma)
```

19.1.10.1 Multiple Mediators: Structural Equation Modeling Approach

A popular method for estimating multiple mediation models is **Structural Equation Modeling** using **lavaan**.

To test multiple mediation, we first **simulate data** where two mediators (M_1 and M_2) contribute to the outcome (Y).

```
# Load required packages
library(MASS)  # For mvrnorm (generating correlated errors)
library(lavaan)

# Function to generate synthetic data
generate_data <- function(n = 10000, a1 = 0.5, a2 = -0.35,
                          b1 = 0.7, b2 = 0.48,
                          corr = TRUE, correlation_value = 0.7) {
```

```
    set.seed(12345)
    X <- rnorm(n)  # Independent variable

    # Generate correlated errors for mediators
    if (corr) {
        Sigma <- matrix(c(1, correlation_value, correlation_value, 1), nrow = 2)
        errors <- mvrnorm(n, mu = c(0, 0), Sigma = Sigma)
    } else {
        errors <- mvrnorm(n, mu = c(0, 0), Sigma = diag(2))
    }

    M1 <- a1 * X + errors[, 1]
    M2 <- a2 * X + errors[, 2]
    Y  <- b1 * M1 + b2 * M2 + rnorm(n)  # Outcome variable

    return(data.frame(X = X, M1 = M1, M2 = M2, Y = Y))
}
```

We analyze the indirect effects through both mediators (M_1 and M_2).

1. Correctly Modeling Correlated Mediators

```
# Generate data with correlated mediators
Data_corr <- generate_data(n = 10000, corr = TRUE, correlation_value = 0.7)

# Define SEM model for multiple mediation
model_corr <- '
  Y ~ b1 * M1 + b2 * M2 + c * X
  M1 ~ a1 * X
  M2 ~ a2 * X
  M1 ~~ M2  # Correlated mediators (modeling correlation correctly)
'

# Fit SEM model
fit_corr <- sem(model_corr, data = Data_corr)

# Extract parameter estimates
parameterEstimates(fit_corr)[, c("lhs", "rhs", "est", "se", "pvalue")]
#>    lhs rhs    est    se pvalue
#> 1    Y  M1  0.700 0.014  0.000
#> 2    Y  M2  0.487 0.014  0.000
#> 3    Y   X -0.009 0.015  0.545
#> 4   M1   X  0.519 0.010  0.000
#> 5   M2   X -0.340 0.010  0.000
#> 6   M1  M2  0.677 0.012  0.000
#> 7    Y   Y  0.975 0.014  0.000
```

```
#> 8   M1  M1  0.973 0.014  0.000
#> 9   M2  M2  0.982 0.014  0.000
#> 10   X   X  1.000 0.000     NA
```

2. Incorrectly Ignoring Correlation Between Mediators

```
# Define SEM model without modeling mediator correlation
model_uncorr <- '
  Y ~ b1 * M1 + b2 * M2 + c * X
  M1 ~ a1 * X
  M2 ~ a2 * X
'

# Fit incorrect model
fit_uncorr <- sem(model_uncorr, data = Data_corr)

# Compare parameter estimates
parameterEstimates(fit_uncorr)[, c("lhs", "rhs", "est", "se", "pvalue")]
#>   lhs rhs    est    se pvalue
#> 1   Y  M1  0.700 0.010  0.000
#> 2   Y  M2  0.487 0.010  0.000
#> 3   Y   X -0.009 0.012  0.443
#> 4  M1   X  0.519 0.010  0.000
#> 5  M2   X -0.340 0.010  0.000
#> 6   Y   Y  0.975 0.014  0.000
#> 7  M1  M1  0.973 0.014  0.000
#> 8  M2  M2  0.982 0.014  0.000
#> 9   X   X  1.000 0.000     NA
```

Comparison of Model Fits

To check whether modeling correlation matters, we compare AIC and RMSEA.

```
# Extract model fit statistics
fit_measures <- function(fit) {
  fitMeasures(fit, c("aic", "bic", "rmsea", "chisq"))
}

# Compare model fits
fit_measures(fit_corr)  # Correct model (correlated mediators)
#>      aic      bic    rmsea    chisq
#> 77932.45 77997.34     0.00     0.00
fit_measures(fit_uncorr)  # Incorrect model (ignores correlation)
#>       aic       bic     rmsea     chisq
#> 84453.208 84510.891     0.808  6522.762
```

- If AIC and RMSEA are lower in the correlated model, it suggests that accounting for correlated errors improves fit.

After fitting the model, we assess:

1. **Direct Effect**: The effect of X on Y **after accounting for both mediators** (c').
2. **Indirect Effects**:
   - $a_1 \times b_1$: Effect of $X \to M_1 \to Y$.
   - $a_2 \times b_2$: Effect of $X \to M_2 \to Y$.
3. **Total Effect**: Sum of direct and indirect effects.

```
# Extract indirect and direct effects
parameterEstimates(fit_corr, standardized = TRUE)
#>    lhs op rhs label    est    se       z pvalue ci.lower ci.upper std.lv
#> 1    Y  ~  M1    b1  0.700 0.014  50.489  0.000    0.673    0.727  0.700
#> 2    Y  ~  M2    b2  0.487 0.014  35.284  0.000    0.460    0.514  0.487
#> 3    Y  ~   X     c -0.009 0.015  -0.606  0.545   -0.038    0.020 -0.009
#> 4   M1  ~   X    a1  0.519 0.010  52.563  0.000    0.499    0.538  0.519
#> 5   M2  ~   X    a2 -0.340 0.010 -34.314  0.000   -0.360   -0.321 -0.340
#> 6   M1 ~~  M2        0.677 0.012  56.915  0.000    0.654    0.700  0.677
#> 7    Y ~~   Y        0.975 0.014  70.711  0.000    0.948    1.002  0.975
#> 8   M1 ~~  M1        0.973 0.014  70.711  0.000    0.946    1.000  0.973
#> 9   M2 ~~  M2        0.982 0.014  70.711  0.000    0.955    1.010  0.982
#> 10   X ~~   X        1.000 0.000      NA     NA    1.000    1.000  1.000
#>    std.all std.nox
#> 1    0.528   0.528
#> 2    0.345   0.345
#> 3   -0.006  -0.006
#> 4    0.465   0.465
#> 5   -0.325  -0.325
#> 6    0.692   0.692
#> 7    0.447   0.447
#> 8    0.784   0.784
#> 9    0.895   0.895
#> 10   1.000   1.000
```

If c' is reduced (but still significant), we have partial mediation. If $c' \approx 0$, it suggests full mediation.

```
# Load required packages
library(MASS)  # for mvrnorm
library(lavaan)

# Function to generate synthetic data with correctly correlated errors for mediators
generate_data <-
  function(n = 10000,
           a1 = 0.5,
```

```
           a2 = -0.35,
           b1 = 0.7,
           b2 = 0.48,
           corr = TRUE,
           correlation_value = 0.7) {
    set.seed(12345)
    X <- rnorm(n)

    # Generate correlated errors using a multivariate normal distribution
    if (corr) {
        Sigma <- matrix(c(1, correlation_value, correlation_value, 1), nrow = 2)  # Higher covari
        errors <- mvrnorm(n, mu = c(0, 0), Sigma = Sigma)  # Generate correlated errors
    } else {
        errors <- mvrnorm(n, mu = c(0, 0), Sigma = diag(2))  # Independent errors
    }

    M1 <- a1 * X + errors[, 1]
    M2 <- a2 * X + errors[, 2]
    Y <- b1 * M1 + b2 * M2 + rnorm(n)  # Y depends on M1 and M2

    data.frame(X = X, M1 = M1, M2 = M2, Y = Y)
}

# Ground truth for comparison
ground_truth <- data.frame(Parameter = c("b1", "b2"), GroundTruth = c(0.7, 0.48))

# Function to extract relevant estimates, standard errors, and model fit
extract_estimates_b1_b2 <- function(fit) {
    estimates <- parameterEstimates(fit)
    estimates <- estimates[estimates$lhs == "Y" & estimates$rhs %in% c("M1", "M2"), c("rhs", "est
    estimates$Parameter <- ifelse(estimates$rhs == "M1", "b1", "b2")
    estimates <- estimates[, c("Parameter", "est", "se")]
    fit_stats <- fitMeasures(fit, c("aic", "bic", "rmsea", "chisq"))
    return(list(estimates = estimates, fit_stats = fit_stats))
}

# Case 1: Correlated errors for mediators (modeled correctly)
Data_corr <- generate_data(n = 10000, corr = TRUE, correlation_value = 0.7)
model_corr <- '
  Y ~ b1 * M1 + b2 * M2 + c * X
  M1 ~ a1 * X
  M2 ~ a2 * X
  M1 ~~ M2  # Correlated mediators (errors)
'
fit_corr <- sem(model = model_corr, data = Data_corr)
```

```r
results_corr <- extract_estimates_b1_b2(fit_corr)

# Case 2: Uncorrelated errors for mediators (modeled correctly)
Data_uncorr <- generate_data(n = 10000, corr = FALSE)
model_uncorr <- '
  Y ~ b1 * M1 + b2 * M2 + c * X
  M1 ~ a1 * X
  M2 ~ a2 * X
'
fit_uncorr <- sem(model = model_uncorr, data = Data_uncorr)
results_uncorr <- extract_estimates_b1_b2(fit_uncorr)

# Case 3: Correlated errors, but not modeled as correlated
fit_corr_incorrect <- sem(model = model_uncorr, data = Data_corr)
results_corr_incorrect <- extract_estimates_b1_b2(fit_corr_incorrect)

# Case 4: Uncorrelated errors, but modeled as correlated
fit_uncorr_incorrect <- sem(model = model_corr, data = Data_uncorr)
results_uncorr_incorrect <- extract_estimates_b1_b2(fit_uncorr_incorrect)

# Combine all estimates for comparison
estimates_combined <- list(
    "Correlated (Correct)" = results_corr$estimates,
    "Uncorrelated (Correct)" = results_uncorr$estimates,
    "Correlated (Incorrect)" = results_corr_incorrect$estimates,
    "Uncorrelated (Incorrect)" = results_uncorr_incorrect$estimates
)

# Combine all into a single table
comparison_table <- do.call(rbind, lapply(names(estimates_combined), function(case) {
    df <- estimates_combined[[case]]
    df$Case <- case
    df
}))

# Merge with ground truth for final comparison
comparison_table <- merge(comparison_table, ground_truth, by = "Parameter")

# Display the comparison table
comparison_table
#>   Parameter       est          se                      Case GroundTruth
#> 1        b1 0.7002984 0.013870433     Correlated (Correct)        0.70
#> 2        b1 0.6973612 0.009859426   Uncorrelated (Correct)        0.70
#> 3        b1 0.7002984 0.010010367   Correlated (Incorrect)        0.70
#> 4        b1 0.6973612 0.00985963 4 Uncorrelated (Incorrect)        0.70
```

```
#> 5        b2 0.4871118 0.013805615     Correlated (Correct)       0.48
#> 6        b2 0.4868318 0.010009908   Uncorrelated (Correct)       0.48
#> 7        b2 0.4871118 0.009963588   Correlated (Incorrect)       0.48
#> 8        b2 0.4868318 0.010010119 Uncorrelated (Incorrect)       0.48

# Display model fit statistics for each case
fit_stats_combined <- list(
    "Correlated (Correct)" = results_corr$fit_stats,
    "Uncorrelated (Correct)" = results_uncorr$fit_stats,
    "Correlated (Incorrect)" = results_corr_incorrect$fit_stats,
    "Uncorrelated (Incorrect)" = results_uncorr_incorrect$fit_stats
)

fit_stats_combined
#> $`Correlated (Correct)`
#>      aic      bic    rmsea    chisq
#> 77932.45 77997.34     0.00     0.00
#>
#> $`Uncorrelated (Correct)`
#>       aic       bic     rmsea     chisq
#> 84664.312 84721.995     0.000     0.421
#>
#> $`Correlated (Incorrect)`
#>       aic       bic     rmsea     chisq
#> 84453.208 84510.891     0.808  6522.762
#>
#> $`Uncorrelated (Incorrect)`
#>      aic      bic    rmsea    chisq
#> 84665.89 84730.78     0.00     0.00
```

19.1.11 Multiple Treatments in Mediation

In some cases, **multiple independent variables** (X_1, X_2) influence the same mediators. This is called **multiple treatments mediation** (Hayes and Preacher, 2014).

For an example in **PROCESS (SPSS/R)**, see:
Process Mediation with Multiple Treatments.

19.2 Causal Inference Approach to Mediation

Traditional mediation models assume that regression-based estimates provide valid causal inference. However, **causal mediation analysis** (CMA) extends beyond traditional models by explicitly defining mediation in terms of **potential outcomes and counterfactuals**.

19.2.1 Example: Traditional Mediation Analysis

We begin with a classic **three-step mediation approach**.

```
# Load data
myData <- read.csv("data/mediationData.csv")

# Step 1 (Total Effect: X → Y) [No longer required]
model.0 <- lm(Y ~ X, data = myData)
summary(model.0)
#>
#> Call:
#> lm(formula = Y ~ X, data = myData)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -5.0262 -1.2340 -0.3282  1.5583  5.1622
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   2.8572     0.6932   4.122 7.88e-05 ***
#> X             0.3961     0.1112   3.564 0.000567 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.929 on 98 degrees of freedom
#> Multiple R-squared:  0.1147, Adjusted R-squared:  0.1057
#> F-statistic:  12.7 on 1 and 98 DF,  p-value: 0.0005671

# Step 2 (Effect of X on M)
model.M <- lm(M ~ X, data = myData)
summary(model.M)
#>
#> Call:
#> lm(formula = M ~ X, data = myData)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -4.3046 -0.8656  0.1344  1.1344  4.6954
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  1.49952    0.58920   2.545   0.0125 *
#> X            0.56102    0.09448   5.938 4.39e-08 ***
```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.639 on 98 degrees of freedom
#> Multiple R-squared:  0.2646, Adjusted R-squared:  0.2571
#> F-statistic: 35.26 on 1 and 98 DF,  p-value: 4.391e-08

# Step 3 (Effect of M on Y, controlling for X)
model.Y <- lm(Y ~ X + M, data = myData)
summary(model.Y)
#>
#> Call:
#> lm(formula = Y ~ X + M, data = myData)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -3.7631 -1.2393  0.0308  1.0832  4.0055
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   1.9043     0.6055   3.145   0.0022 **
#> X             0.0396     0.1096   0.361   0.7187
#> M             0.6355     0.1005   6.321 7.92e-09 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.631 on 97 degrees of freedom
#> Multiple R-squared:  0.373,  Adjusted R-squared:  0.3601
#> F-statistic: 28.85 on 2 and 97 DF,  p-value: 1.471e-10

# Step 4: Bootstrapping for ACME
library(mediation)
results <-
    mediate(
        model.M,
        model.Y,
        treat = 'X',
        mediator = 'M',
        boot = TRUE,
        sims = 500
    )
summary(results)
#>
#> Causal Mediation Analysis
#>
```

```
#> Nonparametric Bootstrap Confidence Intervals with the Percentile Method
#>
#>                Estimate 95% CI Lower 95% CI Upper p-value
#> ACME             0.3565       0.2119         0.51  <2e-16 ***
#> ADE              0.3396      -0.1750         0.28   0.760
#> Total Effect     0.3961       0.1743         0.64   0.004 **
#> Prop. Mediated   0.9000       0.5042         1.94   0.004 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Sample Size Used: 100
#>
#>
#> Simulations: 500
```

- **Total Effect**: $\hat{c} = 0.3961 \rightarrow$ effect of X on Y **without** controlling for M.
- **Direct Effect (ADE)**: $\hat{c'} = 0.0396 \rightarrow$ effect of X on Y **after** accounting for M.
- **ACME (Average Causal Mediation Effect)**:
  - ACME $= \hat{c} - \hat{c'} = 0.3961 - 0.0396 = 0.3565$
  - Equivalent to **product of paths**: $\hat{a} \times \hat{b} = 0.56102 \times 0.6355 = 0.3565$.

These calculations do not rely on strong causal assumptions. For a causal interpretation, we need a more rigorous framework.

19.2.2 Two Approaches in Causal Mediation Analysis

The `mediation` package (Imai et al., 2010a, Imai et al. (2010b)) enables **causal mediation analysis**. It supports two inference types:

1. **Model-Based Inference**
   - Assumptions:
     - Treatment is randomized (or approximated via matching).
     - Sequential Ignorability: No unobserved confounding in:
       1. Treatment $\rightarrow$ Mediator
       2. Treatment $\rightarrow$ Outcome
       3. Mediator $\rightarrow$ Outcome
     - This assumption is hard to justify in observational studies.
2. **Design-Based Inference**
   - Relies on experimental design to isolate the causal mechanism.

Notation

We follow the standard potential outcomes framework:

- $M_i(t)$ = mediator under treatment condition t
- $T_i \in 0, 1$ = treatment assignment
- $Y_i(t, m)$ = outcome under treatment t and mediator value m
- X_i = observed pre-treatment covariates

The **treatment effect** for an individual i:

$$\tau_i = Y_i(1, M_i(1)) - Y_i(0, M_i(0))$$

which decomposes into:

1. **Causal Mediation Effect (ACME)**:

$$\delta_i(t) = Y_i(t, M_i(1)) - Y_i(t, M_i(0))$$

2. **Direct Effect (ADE)**:

$$\zeta_i(t) = Y_i(1, M_i(1)) - Y_i(0, M_i(0))$$

Summing up:

$$\tau_i = \delta_i(t) + \zeta_i(1 - t)$$

Sequential Ignorability Assumption

For CMA to be valid, we assume:

$$\{Y_i(t', m), M_i(t)\} \perp T_i | X_i = x$$
$$Y_i(t', m) \perp M_i(t) | T_i = t, X_i = x$$

- First condition is the standard strong ignorability condition where treatment assignment is random conditional on pre-treatment confounders.
- Second condition is stronger where the mediators is also random given the observed treatment and pre-treatment confounders. This condition is satisfied only when there is no unobserved pre-treatment confounders, and post-treatment confounders, and multiple mediators that are correlated.

Key Challenge

Sequential Ignorability is not testable. Researchers should conduct **sensitivity analysis**.

We now fit a **causal mediation model** using `mediation`.

```r
library(mediation)
set.seed(2014)
data("framing", package = "mediation")

# Step 1: Fit mediator model (M ~ T, X)
med.fit <-
    lm(emo ~ treat + age + educ + gender + income, data = framing)

# Step 2: Fit outcome model (Y ~ M, T, X)
out.fit <-
    glm(
        cong_mesg ~ emo + treat + age + educ + gender + income,
        data = framing,
        family = binomial("probit")
    )

# Step 3: Causal Mediation Analysis (Quasi-Bayesian)
med.out <-
    mediate(
        med.fit,
        out.fit,
        treat = "treat",
        mediator = "emo",
        robustSE = TRUE,
        sims = 100
    )  # Use sims = 10000 in practice
summary(med.out)
#>
#> Causal Mediation Analysis
#>
#> Quasi-Bayesian Confidence Intervals
#>
#>                          Estimate 95% CI Lower 95% CI Upper p-value
#> ACME (control)             0.0791       0.0351         0.15  <2e-16 ***
#> ACME (treated)             0.0804       0.0367         0.16  <2e-16 ***
#> ADE (control)              0.0206      -0.0976         0.12    0.70
#> ADE (treated)              0.0218      -0.1053         0.12    0.70
#> Total Effect               0.1009      -0.0497         0.23    0.14
#> Prop. Mediated (control)   0.6946      -6.3109         3.68    0.14
#> Prop. Mediated (treated)   0.7118      -5.7936         3.50    0.14
#> ACME (average)             0.0798       0.0359         0.15  <2e-16 ***
#> ADE (average)              0.0212      -0.1014         0.12    0.70
#> Prop. Mediated (average)   0.7032      -6.0523         3.59    0.14
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#>
#> Sample Size Used: 265
#>
#>
#> Simulations: 100
```

Alternative: Nonparametric Bootstrap

```
med.out <-
    mediate(
        med.fit,
        out.fit,
        boot = TRUE,
        treat = "treat",
        mediator = "emo",
        sims = 100,
        boot.ci.type = "bca"
    )
summary(med.out)
#>
#> Causal Mediation Analysis
#>
#> Nonparametric Bootstrap Confidence Intervals with the BCa Method
#>
#>                           Estimate 95% CI Lower 95% CI Upper p-value
#> ACME (control)              0.0848       0.0424         0.14  <2e-16 ***
#> ACME (treated)              0.0858       0.0410         0.14  <2e-16 ***
#> ADE (control)               0.0117      -0.0726         0.13    0.58
#> ADE (treated)               0.0127      -0.0784         0.14    0.58
#> Total Effect                0.0975       0.0122         0.25    0.06 .
#> Prop. Mediated (control)    0.8698       1.7460       151.20    0.06 .
#> Prop. Mediated (treated)    0.8804       1.6879       138.91    0.06 .
#> ACME (average)              0.0853       0.0434         0.14  <2e-16 ***
#> ADE (average)               0.0122      -0.0756         0.14    0.58
#> Prop. Mediated (average)    0.8751       1.7170       145.05    0.06 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Sample Size Used: 265
#>
#>
#> Simulations: 100
```

If we suspect **moderation**, we include an **interaction term**.

```
med.fit <-
    lm(emo ~ treat + age + educ + gender + income, data = framing)
```

```r
out.fit <-
    glm(
        cong_mesg ~ emo * treat + age + educ + gender + income,
        data = framing,
        family = binomial("probit")
    )

med.out <-
    mediate(
        med.fit,
        out.fit,
        treat = "treat",
        mediator = "emo",
        robustSE = TRUE,
        sims = 100
    )

summary(med.out)
#>
#> Causal Mediation Analysis
#>
#> Quasi-Bayesian Confidence Intervals
#>
#>                            Estimate 95% CI Lower 95% CI Upper p-value
#> ACME (control)              0.07417      0.02401         0.14  <2e-16 ***
#> ACME (treated)              0.09496      0.02702         0.16  <2e-16 ***
#> ADE (control)              -0.01353     -0.11855         0.11    0.76
#> ADE (treated)               0.00726     -0.11007         0.11    0.90
#> Total Effect                0.08143     -0.05646         0.19    0.26
#> Prop. Mediated (control)    0.64510    -14.31243         3.13    0.26
#> Prop. Mediated (treated)    0.98006    -17.83202         4.01    0.26
#> ACME (average)              0.08457      0.02738         0.15  <2e-16 ***
#> ADE (average)              -0.00314     -0.11457         0.12    1.00
#> Prop. Mediated (average)    0.81258    -16.07223         3.55    0.26
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Sample Size Used: 265
#>
#>
#> Simulations: 100
test.TMint(med.out, conf.level = .95)  # Tests for interaction effect
#>
#>  Test of ACME(1) - ACME(0) = 0
#>
```

```
#> data:  estimates from med.out
#> ACME(1) - ACME(0) = 0.020796, p-value = 0.3
#> alternative hypothesis: true ACME(1) - ACME(0) is not equal to 0
#> 95 percent confidence interval:
#>  -0.01757310  0.07110837
```

Since **sequential ignorability** is untestable, we examine how unmeasured confounding affects ACME estimates.

```
# Load required package
library(mediation)

# Simulate some example data
set.seed(123)
n <- 100
data <- data.frame(
  treat = rbinom(n, 1, 0.5),        # Binary treatment
  med = rnorm(n),                    # Continuous mediator
  outcome = rnorm(n)                  # Continuous outcome
)

# Fit the mediator model (med ~ treat)
med_model <- lm(med ~ treat, data = data)

# Fit the outcome model (outcome ~ treat + med)
outcome_model <- lm(outcome ~ treat + med, data = data)

# Perform mediation analysis
med_out <- mediate(med_model,
                   outcome_model,
                   treat = "treat",
                   mediator = "med",
                   sims = 100)

# Conduct sensitivity analysis
sens_out <- medsens(med_out, sims = 100)

# Print and plot results
summary(sens_out)
#>
#> Mediation Sensitivity Analysis for Average Causal Mediation Effect
#>
#> Sensitivity Region
#>
#>        Rho    ACME 95% CI Lower 95% CI Upper R^2_M*R^2_Y* R^2_M~R^2_Y~
#>  [1,] -0.9 -0.6194      -1.3431       0.1043          0.81       0.7807
```

```
#>  [2,] -0.8 -0.3898      -0.8479       0.0682          0.64       0.6168
#>  [3,] -0.7 -0.2790      -0.6096       0.0516          0.49       0.4723
#>  [4,] -0.6 -0.2067      -0.4552       0.0418          0.36       0.3470
#>  [5,] -0.5 -0.1525      -0.3406       0.0355          0.25       0.2409
#>  [6,] -0.4 -0.1083      -0.2487       0.0321          0.16       0.1542
#>  [7,] -0.3 -0.0700      -0.1723       0.0323          0.09       0.0867
#>  [8,] -0.2 -0.0354      -0.1097       0.0389          0.04       0.0386
#>  [9,] -0.1 -0.0028      -0.0648       0.0591          0.01       0.0096
#> [10,]  0.0  0.0287      -0.0416       0.0990          0.00       0.0000
#> [11,]  0.1  0.0603      -0.0333       0.1538          0.01       0.0096
#> [12,]  0.2  0.0928      -0.0317       0.2173          0.04       0.0386
#> [13,]  0.3  0.1275      -0.0333       0.2882          0.09       0.0867
#> [14,]  0.4  0.1657      -0.0369       0.3684          0.16       0.1542
#> [15,]  0.5  0.2100      -0.0422       0.4621          0.25       0.2409
#> [16,]  0.6  0.2642      -0.0495       0.5779          0.36       0.3470
#> [17,]  0.7  0.3364      -0.0601       0.7329          0.49       0.4723
#> [18,]  0.8  0.4473      -0.0771       0.9717          0.64       0.6168
#> [19,]  0.9  0.6768      -0.1135       1.4672          0.81       0.7807
#>
#> Rho at which ACME = 0: -0.1
#> R^2_M*R^2_Y* at which ACME = 0: 0.01
#> R^2_M~R^2_Y~ at which ACME = 0: 0.0096
plot(sens_out)
```

- If ACME confidence intervals contain 0, the effect is not robust to confounding.

Alternatively, using R^2 interpretation, we need to specify the direction of confounder that affects the mediator and outcome variables in `plot` using `sign.prod = "positive"` (i.e., same direction) or `sign.prod = "negative"` (i.e., opposite direction).

```
plot(sens.out, sens.par = "R2", r.type = "total", sign.prod = "positive")
```

Summary: Causal Mediation vs. Traditional Mediation

Aspect	**Traditional Mediation**	**Causal Mediation**
Model Assumption	Linear regressions	Potential outcomes framework
Assumptions Needed	No omitted confounders	Sequential ignorability
Inference Method	Product of coefficients	Counterfactual reasoning
Bootstrapping?	Common	Essential
Sensitivity Analysis?	Rarely used	Strongly recommended

Chapter 20

Prediction and Estimation

In modern statistics, econometrics, and machine learning, two primary goals often motivate data analysis:

1. **Prediction**: To build a function $\hat{f}$ that **accurately predicts** an outcome Y from observed features (predictors) X.
2. **Estimation or Causal Inference**: To **uncover and quantify** the relationship (often causal) between X and Y, typically by estimating parameters like β in a model $Y = g(X; \beta)$.

These goals, while superficially similar, rest on distinct philosophical and mathematical foundations. Below, we explore the difference in detail, illustrating key ideas with formal definitions, theorems, proofs (where relevant), and references to seminal works.

20.1 Conceptual Framing

20.1.1 Predictive Modeling

Predictive modeling focuses on building a function $\hat{f} : \mathcal{X} \to \mathcal{Y}$ that maps inputs X to outputs Y. For simplicity, assume:

- $X \in \mathbb{R}^p$ (though in practice X can be images, text, time series, etc.).
- $Y \in \mathbb{R}$ for regression or $Y \in \{0, 1\}$ (or other finite set) for classification.

The yardstick for success is the function's accuracy in out-of-sample predictions, often measured by a loss function $L(\hat{y}, y)$. We typically choose $\hat{f}$ to minimize expected loss:

$$\text{(Predictive Problem)} \quad \hat{f} = \arg\min_{f \in \mathcal{F}} \mathbb{E}[L(f(X), Y)],$$

where $\mathcal{F}$ is a class of functions (models) and $\mathbb{E}[\cdot]$ is taken over the joint distribution of (X, Y).

20.1.2 Estimation or Causal Inference

By contrast, estimation or causal inference generally aims to uncover the underlying mechanism: *how does X (or a particular component $T \subseteq X$) cause changes in Y?* The canonical problem is to estimate parameters β in a model $m_\beta(x)$ such that:

$$Y = m_\beta(X) + \varepsilon,$$

or, in linear form,

$$Y = X\beta + \varepsilon.$$

A variety of statistical properties—**consistency, unbiasedness, efficiency, confidence intervals, hypothesis tests**—are relevant here. *Causal* interpretations usually require assumptions beyond typical i.i.d. sampling: unconfoundedness, exogeneity, or random assignment, so that β indeed captures how changes in X cause changes in Y.

Key Distinction:

- **Prediction** does not require that the parameters used in $\hat{f}$ reflect any real-world mechanism. As long as out-of-sample predictive performance is good, the model is deemed successful—even if it's a "black box."
- **Causal inference** demands interpretability in terms of structural or exogenous relationships. The main objective is consistent estimation of the true (or theoretically defined) parameter β, which has an economic, biomedical, or policy interpretation.

20.2 Mathematical Setup

20.2.1 Probability Space and Data

We posit a probability space $(\Omega, \mathcal{F}, P)$ and random variables (X, Y) on it. We typically have an i.i.d. sample $\{(X_i, Y_i)\}_{i=1}^n$ from the true distribution $\mathcal{D}$. Let:

$$(X, Y) \sim \mathcal{D}, \quad (X_i, Y_i) \overset{\text{i.i.d.}}{\sim} \mathcal{D}.$$

In prediction, we train on $\{(X_i, Y_i)\}_{i=1}^n$ to obtain $\hat{f}$, and we evaluate on a test point $(\tilde{X}, \tilde{Y})$ drawn from $\mathcal{D}$. In causal inference, we scrutinize the data generat-

ing process carefully, ensuring that we can identify a causal effect. For example, we may require:

- Potential outcomes $\{Y_i(0), Y_i(1)\}$ for treatment effect settings.
- Unconfoundedness or randomization assumptions.

20.2.2 Loss Functions and Risk

A general framework for both tasks is the risk minimization approach. For a function f, define:

- The population (or expected) risk:

$$\mathcal{R}(f) = \mathbb{E}[L(f(X), Y)].$$

- The empirical risk (on a sample of size n):

$$\hat{\mathcal{R}}_n(f) = \frac{1}{n}\sum_{i=1}^{n} L(f(X_i), Y_i).$$

Prediction: We often solve the empirical risk minimization (ERM) problem:

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \hat{\mathcal{R}}_n(f),$$

possibly with regularization. The measure of success is $\mathcal{R}(\hat{f})$, i.e., how well $\hat{f}$ generalizes beyond the training sample.

Causal/Parameter Estimation: We might define an M-estimator for β (Newey and McFadden, 1994). Consider a function $\psi(\beta; X, Y)$ such that the true parameter β_0 satisfies:

$$\mathbb{E}[\psi(\beta_0; X, Y)] = 0.$$

The empirical M-estimator solves

$$\hat{\beta} = \arg\min_{\beta} \left\| \frac{1}{n}\sum_{i=1}^{n} \psi(\beta; X_i, Y_i) \right\|,$$

or equivalently sets it to zero in a method-of-moments sense:

$$\frac{1}{n}\sum_{i=1}^{n} \psi(\hat{\beta}; X_i, Y_i) = 0.$$

Properties like consistency ($\hat{\beta} \xrightarrow{p} \beta_0$) or asymptotic normality ($\sqrt{n}(\hat{\beta} - \beta_0) \xrightarrow{d} N(0, \Sigma)$) are central. The emphasis is on *uncovering the true* β_0 rather than purely predictive accuracy.

20.3 Prediction in Detail

20.3.1 Empirical Risk Minimization and Generalization

In supervised learning, the goal is to find a function f from a class of candidate models $\mathcal{F}$ (e.g., linear models, neural networks, tree-based models) that accurately predicts an outcome Y given an input X. This is typically formulated as an **Empirical Risk Minimization problem**, where we seek to minimize the average loss over the training data:

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(f(X_i), Y_i).$$

where $L(\cdot, \cdot)$ is a **loss function** that quantifies the error between predictions and actual values. Common choices include:

- **Squared Error (Regression):** $L(\hat{y}, y) = (\hat{y} - y)^2$.
- **Absolute Error (Regression):** $L(\hat{y}, y) = |\hat{y} - y|$.
- **Logistic Loss (Classification):** $L(\hat{p}, y) = -[y \log \hat{p} + (1 - y) \log(1 - \hat{p})]$.

By minimizing empirical risk, we find a function $\hat{f}$ that best fits the observed data. However, **minimizing training error does not guarantee good generalization**—the ability of $\hat{f}$ to perform well on unseen data.

20.3.1.1 Overfitting and Regularization

If $\mathcal{F}$ is very large or expressive (e.g., deep neural networks with millions of parameters), $\hat{f}$ can become too **complex**, learning patterns that exist in the training set but do not generalize to new data. This is called **overfitting**.

To mitigate overfitting, we introduce **regularization**, modifying the optimization objective to penalize complex models:

$$\hat{f}_\lambda = \arg\min_{f \in \mathcal{F}} \left\{ \hat{\mathcal{R}}_n(f) + \lambda \Omega(f) \right\}.$$

where:

- $\hat{\mathcal{R}}_n(f) = \frac{1}{n} \sum_{i=1}^{n} L(f(X_i), Y_i)$ is the empirical risk.
- $\Omega(f)$ is a **complexity penalty** that discourages overly flexible models.
- λ controls the strength of regularization.

Common choices of $\Omega(f)$ include:

- **LASSO penalty:** $\|\beta\|_1$ (sparsity constraint in linear models).

- **Ridge penalty:** $\|\beta\|_2^2$ (shrinking coefficients to reduce variance).
- **Neural network weight decay:** $\sum w^2$ (prevents exploding weights).

Regularization encourages **simpler models**, which are more likely to generalize well.

20.3.1.2 Generalization and Statistical Learning Theory

A fundamental question in machine learning is: *How well does* $\hat{f}$ perform on unseen data? This is captured by the **expected risk**:

$$R(f) = \mathbb{E}[L(f(X), Y)].$$

Ideally, we want to minimize the gap between the **true risk** $R(\hat{f})$ and the best possible risk $R(f^*)$ within $\mathcal{F}$:

$$R(\hat{f}) - \min_{f \in \mathcal{F}} R(f).$$

This difference, called the **excess risk**, measures how well $\hat{f}$ generalizes beyond the training sample. **Statistical Learning Theory** provides theoretical tools to analyze this gap (Vapnik, 2013; Hastie et al., 2009). In particular, it establishes generalization bounds that depend on the **capacity** of the function class $\mathcal{F}$.

20.3.1.3 Complexity Measures

Two important ways to quantify the complexity of $\mathcal{F}$ are

- VC Dimension
- Rademacher Complexity

20.3.1.3.1 VC Dimension The VC dimension measures the ability of a hypothesis class $\mathcal{F}$ to fit arbitrary labels. Formally, the VC dimension of $\mathcal{F}$, denoted as $\mathrm{VC}(\mathcal{F})$, is the largest number of points that can be shattered by some function in $\mathcal{F}$.

- A set of points is shattered by $\mathcal{F}$ if, for every possible labeling of these points, there exists a function $f \in \mathcal{F}$ that perfectly classifies them.

Example 1: Linear Classifiers in 2D

- Consider a set of points in $\mathbb{R}^2$ (the plane).
- If $\mathcal{F}$ consists of linear decision boundaries, we can shatter at most three points in general position (because a single line can separate them in any way).

- However, four points cannot always be shattered (e.g., if arranged in an XOR pattern). - Thus, the VC dimension of linear classifiers in $\mathbb{R}^2$ is 3.

Key Property:

- A higher VC dimension means a more expressive model class (higher capacity).
- If $\mathrm{VC}(\mathcal{F})$ is too large, the model can memorize the training set, leading to poor generalization.

20.3.1.3.2 Rademacher Complexity VC dimension is a combinatorial measure, but Rademacher complexity is a more refined, data-dependent measure of function class flexibility.

Intuition: Rademacher complexity quantifies how well functions in $\mathcal{F}$ can **correlate with random noise**. If a function class can fit random labels well, it is too flexible and likely to overfit.

Definition:
Given n training samples, let $\sigma_1, \dots, \sigma_n$ be **independent Rademacher variables** (i.e., random variables taking values ± 1 with equal probability). The **empirical Rademacher complexity** of $\mathcal{F}$ is:

$$\hat{\mathcal{R}}_n(\mathcal{F}) = \mathbb{E}_\sigma \left[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i f(X_i) \right].$$

Interpretation:

- If $\hat{\mathcal{R}}_n(\mathcal{F})$ is large, then $\mathcal{F}$ can fit random noise well $\Rightarrow$ high risk of overfitting.
- If $\hat{\mathcal{R}}_n(\mathcal{F})$ is small, then $\mathcal{F}$ is more stable $\Rightarrow$ better generalization.

Example 2: Linear Models with Bounded Norm

- Suppose $\mathcal{F}$ consists of linear models $f(X) = w^\top X$, where $\|w\| \leq C$.
- The **Rademacher complexity** of this class scales as $\mathcal{O}(C/\sqrt{n})$.
- This suggests that controlling the norm of w (e.g., via Ridge Regression) improves generalization.

20.3.2 Bias-Variance Decomposition

For a regression problem with squared-error loss, a classic decomposition is:

$$\mathbb{E}_{\text{train}}[(\hat{f}(X) - Y)^2] = \underbrace{(\mathbb{E}[\hat{f}(X)] - f^*(X))^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}[(\hat{f}(X) - \mathbb{E}[\hat{f}(X)])^2]}_{\text{Variance}} + \underbrace{\sigma_\varepsilon^2}_{\text{Irreducible Error}}$$

where $f^*(X) = \mathbb{E}[Y \mid X]$. Minimizing the sum of bias^2 and variance is key.

In prediction, a small increase in bias is often acceptable if it yields a large reduction in variance—this can improve out-of-sample performance. However, for causal inference, any added bias is problematic if it distorts the interpretation of parameters.

20.3.3 Example: Linear Regression for Prediction

Consider a linear predictor:

$$\hat{y} = x^\top \hat{\beta}.$$

We choose $\hat{\beta}$ to minimize:

$$\sum_{i=1}^{n} (y_i - x_i^\top \beta)^2 \quad \text{or with a penalty:} \quad \sum_{i=1}^{n} (y_i - x_i^\top \beta)^2 + \lambda \|\beta\|_2^2.$$

Goal: Achieve minimal prediction error on unseen data $(\tilde{x}, \tilde{y})$.

The estimated $\hat{\beta}$ might be biased if we use regularization (e.g., ridge). But from a purely predictive lens, that bias can be advantageous if it lowers variance substantially and thus lowers expected prediction error.

20.3.4 Applications in Economics

In economics (and related social sciences), **prediction** plays an increasingly prominent role (Mullainathan and Spiess, 2017; Athey and Imbens, 2019):

- **Measure Variables**: Predicting missing or proxy variables (e.g., predicting income from observable covariates, or predicting individual preferences from online behaviors).
- **Embed Prediction Tasks Within Parameter Estimation or Treatment Effects**: Sometimes, a first-stage prediction (e.g., imputing missing data or generating prognostic scores) is used as an input for subsequent causal analyses.
- **Control for Observed Confounders**: Machine learning methods—such as LASSO, random forests, or neural nets—can be used to control for high-dimensional X when doing partial-out adjustments or residualizing outcomes (Belloni et al., 2014; Chernozhukov et al., 2018).

20.4 Parameter Estimation and Causal Inference

20.4.1 Estimation in Parametric Models

In a simple parametric form:

$$Y = X\beta + \varepsilon, \quad \mathbb{E}[\varepsilon \mid X] = 0, \quad \mathrm{Var}(\varepsilon \mid X) = \sigma^2 I.$$

The Ordinary Least Squares estimator is:

$$\hat{\beta}_{\mathrm{OLS}} = \arg\min_{\beta} \|Y - X\beta\|_2^2 = (X^\top X)^{-1} X^\top Y.$$

Under classical assumptions (e.g., no perfect collinearity, homoskedastic errors), $\hat{\beta}_{\mathrm{OLS}}$ is BLUE—the Best Linear Unbiased Estimator.

In a more general form, **parameter estimation**, denoted $\hat{\beta}$, focuses on **estimating the relationship** between y and x, often with a view toward **causality**. In many econometric or statistical settings, we write:

$$y = x^\top \beta + \varepsilon,$$

or more generally $y = g(x; \beta) + \varepsilon$, where β encodes the structural or causal parameters we wish to recover.

The core aim is **consistency**—that is, for large n, we want $\hat{\beta}$ to converge to the true β that defines the underlying relationship. In other words:

$$\hat{\beta} \xrightarrow{p} \beta, \quad \text{as } n \to \infty.$$

Some texts phrase it informally as requiring that

$$\mathbb{E}[\hat{f}] = f,$$

meaning the estimator is (asymptotically) unbiased for the true function or parameters.

However, **consistency** alone may not suffice for scientific inference. One often also examines:

- **Asymptotic Normality**: $\sqrt{n}(\hat{\beta} - \beta) \xrightarrow{d} \mathcal{N}(0, \Sigma)$.
- **Confidence Intervals**: $\hat{\beta}_j \pm z_{\alpha/2}\,\mathrm{SE}(\hat{\beta}_j)$.
- **Hypothesis Tests**: $H_0 \colon \beta_j = 0$ vs. $H_1 \colon \beta_j \neq 0$.

20.4.2 Causal Inference Fundamentals

To interpret β in $Y = X\beta + \varepsilon$ as "causal," we typically require that changes in X (or at least in one component of X) lead to changes in Y that are not confounded by omitted variables or simultaneity. In a prototypical potential-outcomes framework (for a binary treatment D):

- $Y_i(1)$: outcome if unit i receives treatment $D = 1$.
- $Y_i(0)$: outcome if unit i receives no treatment $D = 0$.

The observed outcome Y_i is

$$Y_i = D_i Y_i(1) + (1 - D_i) Y_i(0).$$

The **Average Treatment Effect** (ATE) is:

$$\tau = \mathbb{E}[Y(1) - Y(0)].$$

Identification of τ requires an assumption like unconfoundedness:

$$\{Y(0), Y(1)\} \perp D \mid X,$$

i.e., after conditioning on X, the treatment assignment is as-if random. Estimation strategies then revolve around properly adjusting for X.

Such assumptions are not necessary for raw prediction of Y: a black-box function can yield $\hat{Y} \approx Y$ without ensuring that $\hat{Y}(1) - \hat{Y}(0)$ is an unbiased estimate of τ.

20.4.3 Role of Identification

Identification means that the parameter of interest (β or τ) is uniquely pinned down by the distribution of observables (under assumptions). If β is not identified (e.g., because of endogeneity or insufficient variation in X), no matter how large the sample, we cannot estimate β consistently.

In prediction, "identification" is not usually the main concern. The function $\hat{f}(x)$ could be a complicated ensemble method that just fits well, without guaranteeing any structural or causal interpretation of its parameters.

20.4.4 Challenges

1. **High-Dimensional Spaces**: With large p (number of predictors), covariance among variables (multicollinearity) can hamper classical estimation. This is the setting of the well-known **bias-variance tradeoff** (Hastie et al., 2009; Bishop and Nasrabadi, 2006).

2. **Endogeneity**: If x is correlated with the error term ε, ordinary least squares (OLS) is biased. Causal inference demands identifying exogenous variation in x, which requires additional assumptions or designs (e.g., randomization).
3. **Model Misspecification**: If the functional form $g(x;\beta)$ is incorrect, parameter estimates can systematically deviate from capturing the true underlying mechanism.

20.5 Causation versus Prediction

Understanding the relationship between **causation** and **prediction** is crucial in statistical modeling. Building on Kleinberg et al. (2015) and Mullainathan and Spiess (2017), consider a scenario where Y is an outcome variable dependent on X, and we want to **manipulate** X to maximize some payoff function $\pi(X,Y)$. Formally:

$$\pi(X,Y) = \mathbb{E}\big[U(X,Y)\big] \quad \text{or some other objective measure.}$$

The decision on X depends on **how changes in** X influence π. Taking a derivative:

$$\frac{d\,\pi(X,Y)}{dX} = \frac{\partial \pi}{\partial X}(Y) + \frac{\partial \pi}{\partial Y}\frac{\partial Y}{\partial X}.$$

We can interpret the terms:

- $\dfrac{\partial \pi}{\partial X}$: The *direct* dependence of the payoff on X, which can be predicted if we can forecast how π changes with X.
- $\dfrac{\partial Y}{\partial X}$: The *causal* effect of X on Y, essential for understanding how interventions on X shift Y.
- $\dfrac{\partial \pi}{\partial Y}$: The marginal effect of Y on the payoff.

Hence, Kleinberg et al. (2015) frames this distinction as one between **predicting** Y effectively (for instance, "If I observe X, can I guess Y?") versus **managing** or **causing** Y to change via **interventions** on X. Empirically:

- To **predict** Y, we model $\mathbb{E}[Y \mid X]$.
- To **infer causality**, we require identification strategies that isolate exogenous variation in X.

Empirical work in economics, or social science often aims to estimate partial derivatives of structural or reduced-form equations:

- $\dfrac{\partial Y}{\partial X}$: The **causal derivative**; tells us how Y changes if we **intervene** on X.

- $\dfrac{\partial \pi}{\partial X}$: The **effect of** X on payoff, partially mediated by changes in Y.

Without proper identification (e.g., **randomization, instrumental variables, difference-in-differences, or other quasi-experimental designs**), we risk **conflating association** ($\hat{f}$ that predicts Y) with **causation** ($\hat{\beta}$ that truly captures how X shifts Y).

To illustrate these concepts, consider the following directed acyclic graph (DAG):

```r
library(ggdag)
library(dagitty)
library(ggplot2)


# Define the DAG structure with custom coordinates
dag <- dagitty('
dag {
  X0 [pos="0,1"]
  X [pos="1,2"]
  Y [pos="1,1"]
  II [pos="1,0"]

  X0 -> Y
  X0 -> II
  X -> Y
  Y -> II
}
')

# Convert to ggdag format with manual layout
dag_plot <- ggdag(dag) +
    theme_void() +
    geom_text(aes(x = 0.5, y = 1.2, label = "Causation"), size = 4) +
    geom_text(aes(x = 0.3, y = 0.5, label = "Prediction"), size = 4)

# Display the DAG
dag_plot
```

20.6 Illustrative Equations and Mathematical Contrasts

Below, we showcase a few derivations that highlight how predictive modeling vs. causal inference differ in their mathematical structure and interpretation.

20.6.1 Risk Minimization vs. Consistency

Consider a real-valued outcome Y and predictors X. Let $\ell(y, \hat{y})$ be a loss function, and define the Bayes regressor f^* as:

$$f^* = \arg\min_f \mathbb{E}[\ell(Y, f(X))].$$

For squared error loss, the Bayes regressor is $f^*(x) = \mathbb{E}[Y \mid X = x]$.

A learning algorithm tries to approximate f^*. If we parametrize $f_\beta(x) = x^\top \beta$ and do empirical risk minimization with a large enough sample, β converges to the minimizer of:

$$\beta^* = \arg\min_\beta \mathbb{E}[(Y - X^\top \beta)^2].$$

Note that β^* is the solution to $\mathbb{E}[XX^\top]\beta = \mathbb{E}[XY]$. If $\mathrm{Cov}(X, X)$ is invertible, then

$$\beta^* = \mathrm{Cov}(X, X)^{-1}\mathrm{Cov}(X, Y).$$

This β^* is not necessarily the same as the "true" β_0 from a structural equation $Y = X\beta_0 + \varepsilon$ unless $\mathbb{E}[\varepsilon \mid X] = 0$.

From a predictive standpoint, β^* is the best linear predictor in the sense of mean squared error. From a causal standpoint, we want β_0 such that ε is mean-independent of X. If that fails, $\beta^* \neq \beta_0$.

20.6.2 Partial Derivatives vs. Predictions

A powerful way to see the difference is to compare:

- $\frac{\partial}{\partial x} f^*(x)$ – The partial derivative of the best predictor w.r.t. x. This is about how the model's *prediction* changes with x.
- $\frac{\partial}{\partial x} m_\beta(x)$ – The partial derivative of the structural function $m_\beta(\cdot)$. This is about how the *true outcome* Y changes with x, i.e., a causal effect if m_β is indeed structural.

Unless the model was identified and the assumptions hold (exogeneity, no omitted variables, etc.), the partial derivative from a purely predictive model does not represent the causal effect.

In short: "slopes" from a black-box predictive model are not guaranteed to reflect how interventions on X would shift Y.

20.6.3 Example: High-Dimensional Regularization

Suppose we have a large number of predictors p, possibly $p \gg n$. A common approach in both prediction and inference is LASSO:

$$\hat{\beta}_{\text{LASSO}} = \arg\min_{\beta} \left\{ \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i^\top \beta)^2 + \lambda \|\beta\|_1 \right\}.$$

- **Prediction:** Choose λ to optimize out-of-sample MSE. Some bias is introduced in $\hat{\beta}$, but the final model might predict extremely well, especially if many true coefficients are near zero.
- **Causal Estimation:** We must worry about whether the LASSO is shrinking or zeroing out confounders. If a crucial confounder's coefficient is set to zero, the resulting estimate for a treatment variable's coefficient will be biased. Therefore, special procedures (like the double/debiased machine learning approach (Chernozhukov et al., 2018)) are introduced to correct for the selection bias or to do post-selection inference (Belloni et al., 2014).

The mathematics of "best subset" for prediction vs. valid coverage intervals for parameters diverges significantly.

20.6.4 Potential Outcomes Notation

Let $D \in \{0, 1\}$ be a treatment indicator, and define potential outcomes:

$$Y_i(0), Y_i(1).$$

The observed outcome is:

$$Y_i = D_i Y_i(1) + (1 - D_i) Y_i(0).$$

- **Prediction:** One might train a model $\hat{Y} = \hat{f}(X, D)$ to guess Y from (X, D). That model could be a black box with no guarantee that $\hat{Y}(1) - \hat{Y}(0)$ is an unbiased estimate of $Y_i(1) - Y_i(0)$.
- **Causal Inference:** We want to estimate $\mathbb{E}[Y(1) - Y(0)]$ or $\mathbb{E}[Y(1) - Y(0) \mid X = x]$. Identification typically requires $\{Y(0), Y(1)\} \perp D \mid X$, i.e., after conditioning on X, the treatment assignment is as-if random. Under such an assumption, the difference $\hat{f}(x, 1) - \hat{f}(x, 0)$ can be interpreted as a causal effect.

20.7 Extended Mathematical Points

We now delve deeper into some mathematical nuances that are especially relevant when distinguishing between predictive vs. causal modeling.

20.7.1 M-Estimation and Asymptotic Theory

M-Estimators unify many approaches: maximum likelihood, method of moments, generalized method of moments, and quasi-likelihood estimators. Let β_0 be the true parameter and define the population criterion function:

$$Q(\beta) = \mathbb{E}[m(\beta; X, Y)],$$

for some function m. The M-estimator $\hat{\beta}$ solves:

$$\hat{\beta} = \arg\max_{\beta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} m(\beta; X_i, Y_i).$$

(Or $\arg\min$, depending on convention.)

Under regularity conditions (Newey and McFadden, 1994; White, 1980), we have:

- **Consistency:** $\hat{\beta} \xrightarrow{p} \beta_0$.
- **Asymptotic Normality:** $\sqrt{n}(\hat{\beta} - \beta_0) \xrightarrow{d} N(0, \Sigma)$,

where Σ is derived from derivatives of $m(\cdot;\cdot,\cdot)$ and the distribution of (X, Y).

For prediction, such classical asymptotic properties may be of less interest unless we want to build confidence intervals around predictions. For causal inference, the entire enterprise revolves around these properties to ensure valid inference about β_0.

20.7.2 The Danger of Omitted Variables

Consider a structural equation:

$$Y = \beta_1 X_1 + \beta_2 X_2 + \varepsilon, \quad \mathbb{E}[\varepsilon \mid X_1, X_2] = 0.$$

If we ignore X_2 and regress Y on X_1 only, the resulting $\hat{\beta}_1$ can be severely biased:

$$\hat{\beta}_1 = \arg\min_b \sum_{i=1}^{n} (y_i - b\, x_{i1})^2.$$

The expected value of $\hat{\beta}_1$ in large samples is:

$$\beta_1 \; + \; \beta_2 \, \frac{\mathrm{Cov}(X_1, X_2)}{\mathrm{Var}(X_1)}.$$

This extra term $\beta_2 \, \dfrac{\mathrm{Cov}(X_1, X_2)}{\mathrm{Var}(X_1)}$ is the **omitted variables bias**. For **prediction**, omitting X_2 might sometimes be acceptable if X_2 has little incremental predictive value or if we only care about accuracy in some domain. However, for **inference** on β_1, ignoring X_2 invalidates the causal interpretation.

20.7.3 Cross-Validation vs. Statistical Testing

- **Cross-Validation**: Predominantly used in **prediction** tasks. We split the data into training and validation sets, measure out-of-sample error, and select hyperparameters that minimize CV error.
- **Statistical Testing**: Predominantly used in **inference** tasks. We compute test statistics (e.g., t-test, Wald test), form confidence intervals, or test hypotheses about parameters ($H_0 : \beta_j = 0$).

They serve different objectives:

1. **CV** is about **predictive model selection**.
2. **Testing** is about **scientific or policy conclusions** on whether β_j differs from zero (i.e., "Does a particular variable have a causal effect?").

20.8 Putting It All Together: Comparing Objectives

As an overarching illustration, let $\hat{f}$ be any trained predictor (ML model, regression, etc.) and let $\hat{\beta}$ be a parameter estimator from a structural or causal model. Their respective tasks differ:

- **Form of Output**
  - $\hat{f}$ is a *function* from $\mathcal{X} \to \mathcal{Y}$.
  - $\hat{\beta}$ is a *vector of parameters* with theoretical meaning.
- **Criterion**
  - **Prediction**: Minimizes predictive loss $\mathbb{E}[L(Y, \hat{f}(X))]$.
  - **Causal Inference**: Seeks β such that $Y = m_\beta(X)$ is a correct *structural* representation. Minimizes bias in β, or satisfies orthogonality conditions in method-of-moments style, etc.
- **Validity**
  - **Prediction**: Usually validated by out-of-sample experiments or cross-validation.
  - **Estimation**: Validated by theoretical identification arguments, assumptions about exogeneity, randomization, or no omitted confounders.
- **Interpretation**
  - **Prediction**: "$\hat{f}(x)$ is our best guess of Y for new x."
  - **Causal Inference**: "β measures how Y changes if we intervene on X."

20.9 Conclusion

Prediction and **Estimation/Causal Inference** serve distinctly different roles in data analysis:

- **Prediction**: The emphasis is on **predictive accuracy**. The final model $\hat{f}$ may have uninterpretable parameters (e.g., deep neural networks) yet excel at forecasting Y. Bias in parameter estimates is not necessarily problematic if it reduces variance and improves out-of-sample performance.

- **Estimation/Causal Inference**: The emphasis is on obtaining **consistent** and **unbiased** estimates of parameters (β, or a treatment effect τ). We impose stronger assumptions about data collection and the relationship between X and ε. The success criterion is whether $\hat{\beta} \approx \beta_0$ in a formal sense, with valid confidence intervals and robust identification strategies.

  Key Takeaway:

  - If your question is "How do I **predict** Y for new X as accurately

as possible?", you prioritize **prediction**.

- If your question is "How does **changing** X (or assigning treatment D) affect Y in a **causal** sense?", you focus on **estimation** with a fully developed identification strategy.

IV. CAUSAL INFERENCE

Chapter 21

Causal Inference

Throughout our journey into statistical concepts, we've uncovered patterns, relationships, and trends in data. But now, we arrive at one of the most profound questions in all of research and decision-making: **What truly causes what?**

We've all heard the phrase—*correlation is not causation.*

> *Correlation is not causation.*

Just because two things move together doesn't mean one is pulling the strings of the other. Ice cream sales and drowning incidents both rise in the summer, but ice cream isn't to blame.

But what exactly *is* causation? Let's explore.

One of the most insightful books on this topic is *The Book of Why* by Judea Pearl (Pearl and Mackenzie, 2018), which explains the nuances of causal reasoning beautifully. Below is a concise summary of key ideas from Pearl's work, supplemented with insights from econometrics and statistics.

Understanding causal relationships is essential in research, particularly in fields like economics, finance, marketing, and medicine. While statistical methods have traditionally focused on associational reasoning, causal inference allows us to answer **what-if questions** and make decisions based on interventions.

However, one must be aware of the limitations of statistical methods. As discussed throughout this book, relying solely on data without incorporating domain expertise can lead to misleading conclusions. To establish causality, we often need expert judgment, prior research, and rigorous experimental design.

You may have come across amusing examples of **spurious correlations**—such as the famous Tyler Vigen collection, which shows absurd relationships (e.g.,

"the number of Nicholas Cage movies correlates with drowning accidents"). These highlight the danger of mistaking correlation for causation.

Historically, one of the earliest attempts to infer causation using **regression analysis** was by Yule (1899), who investigated the effect of relief policies on poverty. Unfortunately, his analysis suggested that relief policies increased poverty—a misleading conclusion due to unaccounted confounders.

For a long time, statistics was largely a **causality-free discipline**. The field only began addressing causation in the 1920s, when Sewall Wright introduced **path analysis**, a graphical approach to representing causal relationships. However, it wasn't until Judea Pearl's **Causal Revolution** (1990s) that we gained a formal calculus for causation.

Pearl's framework introduced two key innovations:

1. **Causal Diagrams** (Directed Acyclic Graphs) – A graphical representation of cause-and-effect relationships.
2. **A Symbolic Language**: The Do-Operator ($do(X)$) – A mathematical notation for interventions.

Traditional statistics deals with **conditional probabilities**:

$$P(Y|X)$$

This formula tells us the probability of event Y occurring given that event X has occurred. In the context of observed data, $P(Y \mid X)$ reflects the association between X and Y, showing how likely Y is when X happens.

However, causal inference requires a different concept:

$$P(Y|do(X))$$

which describes what happens when we **actively intervene** and set X. The crucial distinction is:

$$P(Y|X) \neq P(Y|do(X))$$

in general, because **passively observing** X is not the same as **actively manipulating** it.

To make causal claims, we need to answer **counterfactual questions**:

> *What would have happened if we had NOT done X?*

This concept is essential in fields like policy evaluation, medicine, and business decision-making.

To build intelligent systems that can reason causally, we need **an inference engine**:

FIGURE I. How an "inference engine" combines data with causal knowledge to produce answers to queries of interest. The dashed box is not part of the engine but is required for building it. Arrows could also be drawn from boxes 4 and 9 to box 1, but I have opted to keep the diagram simple.

Figure 21.1: p. 12 (Pearl and Mackenzie, 2018)

Pearl outlines **three levels of cognitive ability** required for causal learning:

1. **Seeing** – Observing associations in data.
2. **Doing** – Understanding interventions and predicting their outcomes.
3. **Imagining** – Reasoning about counterfactuals.

These levels correspond to The Ladder of Causation.

21.1 The Ladder of Causation

Pearl's *Ladder of Causation* describes **three hierarchical levels of causal reasoning**:

Level	Activity	Questions Answered	Examples
Association	*Seeing*	What is? How does seeing X change my belief in Y?	What does a symptom tell me about a disease?
Intervention	*Doing*	What if? What happens if I intervene and change X?	If I study more, will my test score improve?
Counterfactuals	*Imagining*	Why? What would have happened if X had been different?	If I had quit smoking a year ago, would I be healthier today?

(Adapted from (Pearl, 2019), p. 57)

Each level requires more cognitive ability and data. Classical statistics operates at Level 1 (association), while causal inference enables us to reach Levels 2 and 3.

21.2 The Formal Notation of Causality

A common mistake is defining causation using probability:

$$X \text{ causes } Y \text{ if } P(Y|X) > P(Y).$$

Seeing X (1st level) **doesn't mean** the probability of Y increases.

It could be either that

1. X causes Y, or
2. Z affects both X and Y. We might be able use **control variables** - $P(Y|X, Z = z) > P(Y|Z = z)$. But then the question becomes
   1. How to choose Z?
   2. Did you choose enough Z?
   3. Did you choose the right Z?

Hence, the previous statement is incorrect. The correct causal statement is:

$$P(Y|do(X)) > P(Y).$$

With **causal diagrams** and **do-calculus**, we can formally express interventions and answer questions at the 2nd level (Intervention).

21.3 The 7 Tools of Structural Causal Models

Pearl's *Structural Causal Model (SCM)* framework provides tools for causal inference (Pearl, 2019):

1. **Encoding Causal Assumptions** – Using **causal graphs** for transparency and testability.
2. **Do-Calculus** – Controlling for confounding using the **backdoor criterion**.
3. **Algorithmization of Counterfactuals** – Modeling "what if?" scenarios.
4. **Mediation Analysis** – Understanding direct vs. indirect effects.
5. **External Validity & Adaptability** – Addressing selection bias and domain adaptation.
6. **Handling Missing Data** – Using causal methods to infer missing information.
7. **Causal Discovery** – Learning causal relationships from data using:
   - **d-separation**
   - **Functional decomposition** (Hoyer et al., 2008)
   - **Spontaneous local changes** (Pearl, 2014)

21.4 Simpson's Paradox

Simpson's Paradox is one of the most striking examples of **why causality matters** and why simple statistical associations can be misleading.

21.4.1 What is Simpson's Paradox?

At its core, Simpson's Paradox occurs when:

> A trend observed in an overall population **reverses** when the population is divided into subgroups.

This means that **statistical associations in raw data can be misleading** if important confounding variables are ignored.

21.4.2 Why is this Important?

Understanding Simpson's Paradox is critical in causal inference because:

1. It highlights the danger of naive data analysis – Just looking at overall trends can lead to incorrect conclusions.
2. It emphasizes the importance of confounding variables – We must control for relevant factors before making causal claims.
3. It demonstrates why causal reasoning is necessary – Simply relying on statistical associations ($P(Y|X)$) without considering structural relationships can lead to paradoxical results.

21.4.3 Comparison between Simpson's Paradox and Omitted Variable Bias

Simpson's Paradox occurs when a trend in an overall dataset **reverses** when broken into subgroups. This happens due to **data aggregation issues**, where differences in subgroup sizes distort the overall trend.

While this often resembles **omitted variable bias (OVB)**—where missing confounders lead to misleading conclusions—Simpson's Paradox is not just a causal inference problem. It is a **mathematical phenomenon** that can arise purely from **improper weighting of data**, even in descriptive statistics.

Similarities Between Simpson's Paradox and OVB:

1. Both involve a missing variable:

- In Simpson's Paradox, a key confounding variable (e.g., customer segment) is hidden in the aggregate data, leading to misleading conclusions.
- In OVB, a relevant variable (e.g., seasonality) is missing from the regression model, causing bias.

2. Both distort causal conclusions:

- OVB biases effect estimates by failing to control for confounding.
- Simpson's Paradox flips statistical relationships when controlling for a confounder.

Differences Between Simpson's Paradox and OVB:

1. Not all OVB cases show Simpson's Paradox:

- OVB generally causes bias, but it doesn't always create a reversal of trends.
- Example: If seasonality increases both ad spend and sales, omitting it inflates the ad spend $\rightarrow$ sales relationship but does not necessarily reverse it.

2. Simpson's Paradox can occur even without causal inference:

- Simpson's Paradox is a mathematical/statistical phenomenon that can arise even in purely observational data, not just causal inference.
- It results from data weighting issues, even if causality is not considered.

3. OVB is a model specification issue; Simpson's Paradox is a data aggregation issue:

- OVB occurs in regression models when we fail to include relevant predictors.
- Simpson's Paradox arises from incorrect data aggregation when groups are not properly analyzed separately.

The Right Way to Think About It

- Simpson's Paradox is often *caused* by omitted variable bias, but they are not the same thing.
- OVB is a problem in causal inference models; Simpson's Paradox is a problem in raw data interpretation.

How to Fix These Issues?

- For OVB: Use causal diagrams, add control variables, and use regression adjustments.
- For Simpson's Paradox: Always analyze subgroup-level trends before making conclusions based on aggregate data.
- Bottom line: Simpson's Paradox is often *caused* by omitted variable bias, but it is not just OVB—it is a fundamental issue of misleading data aggregation.

21.4.4 Illustrating Simpson's Paradox: Marketing Campaign Success Rates

Let's explore this paradox using a practical business example.

21.4.4.1 Scenario: Marketing Campaign Performance

Imagine a company running two marketing campaigns, Campaign A and Campaign B, to attract new customers. We analyze which campaign has a higher conversion rate.

21.4.4.2 Step 1: Creating the Data

We will simulate conversion rates for two different customer segments: **High-Value** customers (who typically convert at a higher rate) and **Low-Value** customers (who convert at a lower rate).

```r
# Load necessary libraries
library(dplyr)

# Create a dataset where:
#  - B is better than A in each individual segment.
#  - A turns out better when we look at the overall (aggregated) data.

marketing_data <- data.frame(
  Campaign = c("A", "A", "B", "B"),
  Segment  = c("High-Value", "Low-Value", "High-Value", "Low-Value"),
  Visitors = c(500, 2000, 300, 3000),    # total visitors in each segment
  Conversions = c(290, 170, 180, 270)   # successful conversions
)
```

```r
# Compute segment-level conversion rate
marketing_data <- marketing_data %>%
  mutate(Conversion_Rate = Conversions / Visitors)

# Print the data
print(marketing_data)
#>   Campaign    Segment Visitors Conversions Conversion_Rate
#> 1        A High-Value      500         290           0.580
#> 2        A  Low-Value     2000         170           0.085
#> 3        B High-Value      300         180           0.600
#> 4        B  Low-Value     3000         270           0.090
```

Interpreting This Data

- **Campaign B** in the High-Value segment: $\frac{180}{300} = 60\%$
- **Campaign A** in the High-Value segment: $\frac{290}{500} = 58\%$

  => B is better in the High-Value segment (60% vs 58%).
- **Campaign B** in the Low-Value segment: $\frac{270}{3000} = 9\%$
- **Campaign A** in the Low-Value segment: $\frac{170}{2000} = 8.5\%$

  => B is better in the Low-Value segment (9% vs 8.5%).

Thus, **B** outperforms **A** in each **individual** segment.

21.4.4.3 Step 2: Aggregating Data (Ignoring Customer Segments)

Now, let's calculate the overall conversion rate for each campaign without considering customer segments.

```r
# Compute overall conversion rates for each campaign
overall_rates <- marketing_data %>%
  group_by(Campaign) %>%
  summarise(
    Total_Visitors     = sum(Visitors),
    Total_Conversions  = sum(Conversions),
    Overall_Conversion_Rate = Total_Conversions / Total_Visitors
  )

# Print overall conversion rates
print(overall_rates)
#> # A tibble: 2 x 4
#>   Campaign Total_Visitors Total_Conversions Overall_Conversion_Rate
#>   <chr>             <dbl>             <dbl>                   <dbl>
#> 1 A                  2500               460                   0.184
#> 2 B                  3300               450                   0.136
```

21.4.4.4 Step 3: Observing Simpson's Paradox

Let's determine which campaign appears to have a higher conversion rate.

```
# Identify the campaign with the higher overall conversion rate
best_campaign_overall <- overall_rates %>%
    filter(Overall_Conversion_Rate == max(Overall_Conversion_Rate)) %>%
    select(Campaign, Overall_Conversion_Rate)

print(best_campaign_overall)
#> # A tibble: 1 x 2
#>   Campaign Overall_Conversion_Rate
#>   <chr>                      <dbl>
#> 1 A                          0.184
```

Even though **Campaign B** is **better** in each **segment**, you should see that **Campaign A** has a **higher** aggregated (overall) conversion rate!

21.4.4.5 Step 4: Conversion Rates Within Customer Segments

We now analyze the conversion rates separately for high-value and low-value customers.

```
# Compute conversion rates by customer segment
by_segment <- marketing_data %>%
  select(Campaign, Segment, Conversion_Rate) %>%
  arrange(Segment)

print(by_segment)
#>   Campaign    Segment Conversion_Rate
#> 1        A High-Value           0.580
#> 2        B High-Value           0.600
#> 3        A  Low-Value           0.085
#> 4        B  Low-Value           0.090
```

- In **High-Value**, B > A.
- In **Low-Value**, B > A.

Yet, overall, A > B.

This **reversal** is the hallmark of **Simpson's Paradox**.

21.4.4.6 Step 5: Visualizing the Paradox

To make this clearer, let's visualize the results.

```
library(ggplot2)

# Plot conversion rates by campaign and segment
```

```r
ggplot(marketing_data,
       aes(x = Segment,
           y = Conversion_Rate,
           fill = Campaign)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(
    title = "Simpson's Paradox in Marketing",
    x     = "Customer Segment",
    y     = "Conversion Rate"
  ) +
  theme_minimal()
```

This bar chart reveals that for **both** segments, B's bar is taller (i.e., B's conversion rate is higher). If you only examined segment-level data, you would conclude that B is the superior campaign.

However, if you aggregate the data (ignore segments), you get the opposite conclusion — that **A** is better overall.

21.4.5 Why Does This Happen?

This paradox arises because of a **confounding variable** — in this case, the **distribution of visitors** across segments.

- **Campaign A** has **more** of its traffic in the High-Value segment (where conversions are generally high).
- **Campaign B** has **many** of its visitors in the Low-Value segment.

Because the **volume** of Low-Value visitors in B is extremely large (3000 vs. 2000 for A), it weighs down B's **overall** average more, allowing A's overall rate to exceed B's.

21.4.6 How Does Causal Inference Solve This?

To avoid Simpson's Paradox, we need to move beyond association and use causal analysis:

1. **Use causal diagrams (DAGs) to model relationships**
   - The marketing campaign choice is confounded by customer segment.
   - We must control for the confounding variable.
2. **Use stratification or regression adjustment**
   - Instead of comparing raw conversion rates, we should compare rates within each customer segment.
   - This ensures that confounding factors do not distort results.
3. **Use the do-operator** to simulate interventions
   - Instead of asking $P(\text{Conversion} \mid \text{Campaign})$, ask: $P(\text{Conversion} \mid do(\text{Campaign}))$
   - This estimates what would happen if we randomly assigned campaigns (removing confounding bias).

21.4.7 Correcting Simpson's Paradox with Regression Adjustment

Let's adjust for the confounding variable using **logistic regression**.

```
# Logistic regression adjusting for the Segment
model <- glm(
  cbind(Conversions, Visitors - Conversions) ~ Campaign + Segment,
  family = binomial(),
  data   = marketing_data
)

summary(model)
#>
#> Call:
#> glm(formula = cbind(Conversions, Visitors - Conversions) ~ Campaign +
#>     Segment, family = binomial(), data = marketing_data)
#>
#> Coefficients:
#>                  Estimate Std. Error z value Pr(>|z|)
#> (Intercept)       0.32783    0.07839   4.182 2.89e-05 ***
```

```
#> CampaignB         0.06910    0.08439   0.819    0.413
#> SegmentLow-Value -2.70806    0.08982 -30.151  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 977.473003  on 3  degrees of freedom
#> Residual deviance:   0.012337  on 1  degrees of freedom
#> AIC: 32.998
#>
#> Number of Fisher Scoring iterations: 3
```

This model includes both **Campaign** and **Segment** as predictors, giving a clearer picture of the *true effect* of each campaign on conversion, **after** controlling for differences in segment composition.

21.4.8 Key Takeaways

1. **Simpson's Paradox demonstrates why causal inference is essential.**
   - Aggregated statistics can be misleading due to hidden confounding.
   - Breaking data into subgroups can reverse conclusions.
2. **Causal reasoning helps identify and correct paradoxes.**
   - Using causal graphs, do-calculus, and adjustment techniques, we can find the true causal effect.
3. **Naïve data analysis can lead to bad business decisions.**
   - If a company allocated more budget to Campaign B based on overall conversion rates, it might be investing in the wrong strategy!

21.5 Additional Resources

To explore causal inference in R, check out the CRAN Task View for Causal Inference:

For further reading:

- *The Book of Why* – Judea Pearl (Pearl and Mackenzie, 2018)
- *Causal Inference in Statistics: A Primer* – Pearl, Glymour, Jewell
- *Causality: Models, Reasoning, and Inference* – Judea Pearl

21.6 Experimental vs. Quasi-Experimental Designs

Experimental and quasi-experimental designs differ in their approach to causal inference. The table below summarizes key distinctions:

Experimental Design	**Quasi-Experimental Design**
Conducted by an experimentalist	Conducted by an observationalist
Uses experimental data	Uses observational data
Random assignment reduces treatment imbalance	Random sampling reduces sample selection error

21.6.1 Criticisms of Quasi-Experimental Designs

Quasi-experimental methods do not always approximate experimental results accurately. For instance, LaLonde (1986) demonstrates that commonly used methods such as:

- Matching Methods
- Difference-in-differences
- Tobit-2 (Heckman-type models)

often fail to replicate experimental estimates reliably.

21.7 Hierarchical Ordering of Causal Tools

Causal inference tools can be categorized based on their methodological rigor, with **randomized controlled trials (RCTs)** considered the gold standard.

1. Experimental Design: Randomized Control Trials (Gold standard)
2. Quasi-experimental
    1. Regression Discontinuity
    2. Synthetic Difference-in-Differences
    3. Difference-In-Differences
    4. Synthetic Control
    5. Event Studies
    6. Fixed Effects Estimator
    7. Endogenous Treatment: mostly Instrumental Variables
    8. Matching Methods

9. Interrupted Time Series
10. Endogenous Sample Selection

21.8 Types of Validity in Research

Validity in research includes:

1. Measurement Validity (e.g., construct, content, criterion, face validity)
2. Internal Validity
3. External Validity
4. Ecological Validity
5. Statistical Conclusion Validity

By examining these, you can ensure that your study's measurements are accurate, your findings are reliably causal, and your conclusions generalize to broader contexts.

21.8.1 Measurement Validity

Measurement validity pertains to whether the instrument or method you use truly measures what it's intended to measure. Within this umbrella, there are several sub-types:

21.8.1.1 Face Validity

- **Definition**: The extent to which a measurement or test *appears* to measure what it is supposed to measure, at face value (i.e., does it "look" right to experts or users?).
- **Importance**: While often considered a less rigorous form of validity, it's useful for ensuring the test or instrument is intuitively acceptable to stakeholders, participants, or experts in the field.
- **Example**: A questionnaire measuring "anxiety" that has questions about nervousness, worries, and stress has good face validity because it obviously seems to address anxiety.

21.8.1.2 Content Validity

- **Definition**: The extent to which a test or measurement covers *all* relevant facets of the construct it aims to measure.

- **Importance**: Especially critical in fields like education or psychological testing, where you want to ensure the entire domain of a subject/construct is properly sampled.
- **Example**: A math test that includes questions on algebra, geometry, and calculus might have high content validity for a comprehensive math skill assessment. If it only tested algebra, the content validity would be low.

21.8.2 Construct Validity

- **Definition**: The degree to which a test or measurement tool accurately represents the theoretical construct it intends to measure (e.g., intelligence, motivation, self-esteem).
- **Types of Evidence**:
  - **Convergent Validity**: Demonstrated when measures that are supposed to be related (theoretically) are observed to correlate.
  - **Discriminant (Divergent) Validity**: Demonstrated when measures that are supposed to be unrelated theoretically do not correlate.
- **Example**: A new questionnaire on "job satisfaction" should correlate with other established job satisfaction questionnaires (convergent validity) but should not correlate strongly with unrelated constructs like "physical health" (discriminant validity).

21.8.3 Criterion Validity

- **Definition**: The extent to which the measurement predicts or correlates with an outcome criterion. In other words, do scores on the measure relate to an external standard or "criterion"?
- **Types**:
  - **Predictive Validity**: The measure predicts a future outcome (e.g., an entrance exam predicting college success).
  - **Concurrent Validity**: The measure correlates with an existing, accepted measure taken at the same time (e.g., a new depression scale compared with a gold-standard clinical interview).
- **Example**: A new test of driving skills has high criterion validity if people who score highly perform better on actual road tests (predictive validity).

21.8.4 Internal Validity

Internal validity refers to the extent to which a study can establish a *cause-and-effect* relationship. High internal validity means you can be confident that

the observed effects are due to the treatment or intervention itself and not due to confounding factors or alternative explanations. This is the validity that economists and applied scientists largely care about.

21.8.4.1 Major Threats to Internal Validity

1. **Selection Bias**: Systematic differences between groups that exist before the treatment is applied.
2. **History Effects**: External events occurring during the study can affect outcomes (e.g., economic downturn during a job-training study).
3. **Maturation**: Participants might change over time simply due to aging, learning, fatigue, etc., independent of the treatment.
4. **Testing Effects**: Taking a test more than once can influence participants' responses (practice effect).
5. **Instrumentation**: Changes in the measurement instrument or the observers can lead to inconsistencies in data collection.
6. **Regression to the Mean**: Extreme pre-test scores tend to move closer to the average on subsequent tests.
7. **Attrition (Mortality)**: Participants dropping out of the study in ways that are systematically related to the treatment or outcomes.

21.8.4.2 Strategies to Improve Internal Validity

- **Random Assignment**: Ensures that, on average, groups are equivalent on both known and unknown variables.
- **Control Groups**: Provide a baseline for comparison to isolate the effect of the intervention.
- **Blinding (Single-, Double-, or Triple-blind)**: Reduces biases from participants, researchers, or analysts.
- **Standardized Procedures and Protocols**: Minimizes variability in how interventions or measurements are administered.
- **Matching or Stratification**: When randomization is not possible, matching participants on key characteristics can reduce selection bias.
- **Pretest-Posttest Designs**: Compare participant performance before and after the intervention (though watch for testing effects).

21.8.5 External Validity

External validity addresses the generalizability of the findings beyond the specific context of the study. A study with high external validity can be applied

to other populations, settings, or times. On the other hand, localness can affect external validity.

21.8.5.1 Subtypes (or Related Concepts) of External Validity

1. **Population Validity**: The degree to which study findings can be generalized to the larger population from which the sample was drawn.
2. **Ecological Validity** (sometimes considered separately): Whether findings obtained in controlled conditions can be applied to real-world settings.
3. **Temporal Validity**: Whether the results of the study hold true over time. Changing societal norms, technologies, or economic conditions might render findings obsolete.

21.8.5.2 Threats to External Validity

- **Unrepresentative Samples**: If the sample does not reflect the wider population (in demographics, culture, etc.), generalization is limited.
- **Artificial Research Environments**: Highly controlled lab settings may not capture real-world complexities.
- **Treatment-Setting Interaction**: The effect of the treatment might depend on the unique conditions of the setting (e.g., a particular school, hospital, or region).
- **Treatment-Selection Interaction**: Certain characteristics of the selected participants might interact with the treatment (e.g., results from a specialized population do not apply to the general public).

21.8.5.3 Strategies to Improve External Validity

- **Use of Diverse and Representative Samples**: Recruit participants that mirror the larger population.
- **Field Studies and Naturalistic Settings**: Conduct experiments in real-world environments rather than artificial labs.
- **Replication in Multiple Contexts**: Replicate the study across different settings, geographic locations, and populations.
- **Longitudinal Studies**: Evaluate whether relationships hold over extended periods.

21.8.6 Ecological Validity

Ecological validity is often discussed as a subcategory of external validity. It specifically focuses on the *realism* of the study environment and tasks:

- **Definition**: The degree to which study findings can be generalized to the real-life settings where people actually live, work, and interact.
- **Key Idea**: Even if a lab experiment shows a particular behavior, do people behave the same way in their daily lives with everyday distractions, social pressures, and contextual factors?

21.8.6.1 Enhancing Ecological Validity

- **Naturalistic Observation**: Conduct observations or experiments in participants' usual environments.
- **Realistic Tasks**: Use tasks that closely mimic real-world challenges or behaviors.
- **Minimal Interference**: Researchers strive to reduce the artificiality of the setting, allowing participants to behave as naturally as possible.

21.8.7 Statistical Conclusion Validity

Though often discussed alongside internal validity, **statistical conclusion validity** focuses on whether the statistical tests used in a study are appropriate, powerful enough, and applied correctly.

21.8.7.1 Threats to Statistical Conclusion Validity

1. **Low Statistical Power**: If the sample size is too small, the study may fail to detect a real effect (Type II error).
2. **Violations of Statistical Assumptions**: Incorrect application of statistical tests can lead to spurious conclusions (e.g., using parametric tests with non-normal data without appropriate adjustments).
3. **Fishing and Error Rate Problem**: Running many statistical tests without correction increases the chance of a Type I error (finding a false positive).
4. **Reliability of Measures**: If the measurement instruments are unreliable, statistical correlations or differences may be undervalued or overstated.

21.8.7.2 Improving Statistical Conclusion Validity

- **Adequate Sample Size**: Conduct a power analysis to determine the necessary size to detect meaningful effects.
- **Appropriate Statistical Techniques**: Ensure your chosen analysis matches the nature of the data and research question.
- **Multiple Testing Corrections**: Use methods like Bonferroni or false discovery rate corrections when conducting multiple comparisons.

- **High-Quality Measurements**: Use reliable and valid measures to reduce measurement error.

21.8.8 Putting It All Together

1. **Face Validity**: Does it look like it measures what it should?
2. **Content Validity**: Does it cover all facets of the construct?
3. **Construct Validity**: Does it truly reflect the theoretical concept?
4. **Criterion Validity**: Does it correlate with or predict other relevant outcomes?
5. **Internal Validity**: Is the relationship between treatment and outcome truly causal?
6. **External Validity**: Can findings be generalized to other populations, settings, and times?
7. **Ecological Validity**: Are the findings applicable to real-world scenarios?
8. **Statistical Conclusion Validity**: Are the statistical inferences correct and robust?

Researchers typically need to strike a balance among these different validities:

- A **highly controlled lab study** might excel in internal validity but might have limited external and ecological validity.
- A **broad, naturalistic field study** might have stronger external or ecological validity but weaker internal validity due to less control over confounding variables.

No single study can maximize all validity types simultaneously, so replication, triangulation (using multiple methods), and transparent reporting are crucial strategies to bolster overall credibility.

21.9 Types of Subjects in a Treatment Setting

When conducting causal inference, particularly in randomized experiments or quasi-experimental settings, individuals in the study can be classified into four distinct groups based on their response to treatment assignment. These groups differ in how they react when they are assigned to receive or not receive treatment.

21.9.1 Non-Switchers

Non-switchers are individuals whose treatment status does not change regardless of whether they are assigned to the treatment or control group. These individuals do not provide useful causal information because their behavior remains unchanged. They are further divided into:

- **Always-Takers**: These individuals will **always receive** the treatment, even if they are assigned to the control group.
- **Never-Takers**: These individuals will **never receive** the treatment, even if they are assigned to the treatment group.

Since their behavior is independent of the assignment, always-takers and never-takers do not contribute to identifying causal effects in standard randomized experiments. Instead, their presence can introduce bias in treatment effect estimation, particularly in **intention-to-treat analysis**.

21.9.2 Switchers

Switchers are individuals whose treatment status **depends on the assignment**. These individuals are the primary focus of causal inference because they provide meaningful information about the effect of treatment. They are classified into:

- **Compliers**: Individuals who follow the assigned treatment protocol.
  - If assigned to the treatment group, they **accept and receive** the treatment.
  - If assigned to the control group, they **do not receive** the treatment.
  - **Why are compliers important?**
    * They are the only group for whom treatment assignment affects actual treatment receipt.
    * Causal effect estimates (such as the local average treatment effect, LATE) are typically identified using compliers.
    * If the dataset only contains compliers, then the intention-to-treat effect (ITT) is equal to the treatment effect.
- **Defiers**: Individuals who do the **opposite** of what they are assigned.
  - If assigned to the treatment group, they **refuse the treatment**.
  - If assigned to the control group, they **seek out and receive** the treatment anyway.
  - **Why are defiers typically ignored?**
    * In most studies, defiers are assumed to be a small or negligible group.
    * Standard causal inference frameworks assume **monotonicity**, meaning no one behaves as a defier.
    * If defiers exist in large numbers, estimating causal effects becomes significantly more complex.

21.9.3 Classification of Individuals Based on Treatment Assignment

The following table summarizes how different types of individuals respond to treatment and control assignments:

	Treatment Assignment	Control Assignment
Compliers	Treated	Not Treated
Always-Takers	Treated	Treated
Never-Takers	Not Treated	Not Treated
Defiers	Not Treated	Treated

Key Takeaways:

1. **Compliers** are the only group that allows us to estimate causal effects using **randomized or quasi-experimental designs**.
2. **Always-Takers and Never-Takers** do not provide meaningful variation in treatment status, making them less useful for causal inference.
3. **Defiers** typically violate the assumption of monotonicity, and their presence complicates causal estimation.
4. If a dataset consists **only of compliers**, the **intention-to-treat effect** will be equal to the **treatment effect**.

By correctly identifying and accounting for these different subject types, researchers can ensure more accurate causal inference and minimize biases in estimating treatment effects.

21.10 Types of Treatment Effects

When evaluating the causal impact of an intervention, different estimands (quantities of interest) can be used to measure treatment effects, depending on the study design and assumptions about compliance.

Terminology:

- **Estimands**: The causal effect parameters we seek to measure.
- **Estimators**: The statistical procedures used to estimate those parameters.
- **Sources of Bias** (Keele and Grieve, 2025):

$$\begin{aligned}&\text{Estimator - True Causal Effect}\\&= \underbrace{\textbf{Hidden bias}}_{\text{Due to design}} + \underbrace{\textbf{Misspecification bias}}_{\text{Due to modeling}} + \underbrace{\textbf{Statistical noise}}_{\text{Due to finite sample}}\end{aligned}$$

21.10.1 Average Treatment Effect

The **Average Treatment Effect (ATE)** is the expected difference in outcomes between individuals who receive treatment and those who do not.

Definition

Let:

- $Y_i(1)$ be the outcome of individual i under treatment.
- $Y_i(0)$ be the outcome of individual i under control.

The **individual treatment effect** is:

$$\tau_i = Y_i(1) - Y_i(0)$$

Since we cannot observe both $Y_i(1)$ and $Y_i(0)$ for the same individual (a fundamental problem in causal inference), we estimate the **ATE** across a population:

$$ATE = E[Y(1)] - E[Y(0)]$$

Identification Under Randomization

If treatment assignment is **randomized** (under Experimental Design), then the **observed** difference in means between treatment and control groups provides an **unbiased estimator** of ATE:

$$ATE = \frac{1}{N}\sum_{i=1}^{N}\tau_i = \frac{\sum_1^N Y_i(1)}{N} - \frac{\sum_i^N Y_i(0)}{N}$$

With **randomization**, we assume:

$$E[Y(1)|D=1] = E[Y(1)|D=0] = E[Y(1)]$$

$$E[Y(0)|D=1] = E[Y(0)|D=0] = E[Y(0)]$$

Thus, the **difference in observed means** between treated and control groups provides an **unbiased estimate** of ATE.

$$ATE = E[Y(1)] - E[Y(0)]$$

Alternatively, we can express the **potential outcomes framework** in a regression form, which allows us to connect causal inference concepts with standard regression analysis.

Instead of writing treatment effects as potential outcomes, we can define the observed outcome Y_i in terms of a regression equation:

$$Y_i = Y_i(0) + [Y_i(1) - Y_i(0)]D_i$$

where:

- $Y_i(0)$ is the outcome if individual i **does not** receive treatment.
- $Y_i(1)$ is the outcome if individual i **does** receive treatment.
- D_i is a binary indicator for treatment assignment:
- $D_i = 1$ if individual i receives treatment.
- $D_i = 0$ if individual i is in the control group.

We can redefine this equation using regression notation:

$$Y_i = \beta_{0i} + \beta_{1i}D_i$$

where:

- $\beta_{0i} = Y_i(0)$ represents the **baseline (control group) outcome**.
- $\beta_{1i} = Y_i(1) - Y_i(0)$ represents the **individual treatment effect**.

Thus, in an ideal setting, **the coefficient on** D_i in a regression gives us the treatment effect.

In observational studies, treatment assignment D_i is often **not random**, leading to **endogeneity**. This means that the error term in the regression equation might be correlated with D_i, violating one of the key assumptions of the Ordinary Least Squares estimator.

To formalize this issue, we can express the outcome equation as:

$$\begin{aligned} Y_i &= \beta_{0i} + \beta_{1i}D_i \\ &= (\bar{\beta}_0 + \epsilon_{0i}) + (\bar{\beta}_1 + \epsilon_{1i})D_i \\ &= \bar{\beta}_0 + \epsilon_{0i} + \bar{\beta}_1 D_i + \epsilon_{1i}D_i \end{aligned}$$

where:

- $\bar{\beta}_0$ is the average baseline outcome.

- $\bar{\beta}_1$ is the average treatment effect.
- ϵ_{0i} captures individual-specific deviations in control group outcomes.
- ϵ_{1i} captures heterogeneous treatment effects.

If treatment assignment is truly **random**, then:

$$E[\epsilon_{0i}] = E[\epsilon_{1i}] = 0$$

which ensures:

- **No selection bias**: $D_i \perp \epsilon_{0i}$ (i.e., treatment assignment is independent of the baseline error).
- **Treatment effect is independent of assignment**: $D_i \perp \epsilon_{1i}$.

However, in observational studies, these assumptions often **fail**. This leads to:

- **Selection bias**: If individuals self-select into treatment based on unobserved characteristics, then D_i correlates with ϵ_{0i}.
- **Heterogeneous treatment effects**: If the treatment effect itself varies across individuals, then D_i correlates with ϵ_{1i}.

These issues violate the exogeneity assumption in OLS regression, leading to biased estimates of β_1.

When estimating treatment effects using OLS regression, we need to be aware of potential estimation issues.

1. **OLS Estimator and Difference-in-Means**

Under **random assignment**, the OLS estimator for β_1 simplifies to the **difference in means estimator**:

$$\hat{\beta}_1^{OLS} = \bar{Y}_{\text{treated}} - \bar{Y}_{\text{control}}$$

which is an **unbiased estimator** of the Average Treatment Effect.

However, when treatment assignment is **not random**, OLS estimates may be biased due to unobserved confounders.

2. **Heteroskedasticity and Robust Standard Errors**

If treatment effects vary across individuals (i.e., **treatment effect heterogeneity**), the error term contains an interaction:

$$\epsilon_i = \epsilon_{0i} + D_i \epsilon_{1i}$$

which leads to **heteroskedasticity** (i.e., the variance of errors depends on D_i and possibly on covariates X_i).

To address this, we use **heteroskedasticity-robust standard errors**, which ensure valid inference even when variance is not constant across observations.

21.10.2 Conditional Average Treatment Effect

Treatment effects may **vary across different subgroups** in a population. The **Conditional Average Treatment Effect (CATE)** captures heterogeneity in treatment effects across subpopulations.

Definition

For a subgroup characterized by covariates X_i:

$$CATE = E[Y(1) - Y(0)|X_i]$$

Why is CATE Useful?

- **Heterogeneous Treatment Effects**: Certain groups may benefit more from treatment than others.
- **Policy Targeting**: Understanding who benefits the most allows for better resource allocation.

Example

- Policy Intervention: A job training program may have different effects on younger vs. older workers.
- Medical Treatments: Drug effectiveness may differ by gender, age, or genetic factors.

Estimating CATE allows policymakers and researchers to **identify who benefits most** from an intervention.

21.10.3 Intention-to-Treat Effect

A key issue in empirical research is **non-compliance**, where individuals do not always follow their assigned treatment (i.e., either people who are supposed to receive treatment don't receive it, or people who are supposed to be in the control group receive the treatment). The **Intention-to-Treat (ITT) effect** measures the impact of offering treatment, regardless of whether individuals actually receive it.

Definition

The **ITT effect** is the observed difference in means between groups **assigned** to treatment and control:

$$ITT = E[Y|D = 1] - E[Y|D = 0]$$

Why Use ITT?

- **Policy Evaluation**: ITT reflects the **real-world effectiveness** of an intervention, accounting for incomplete take-up.
- **Randomized Trials**: ITT preserves **randomization**, even when compliance is imperfect.

Example: Vaccination

- A government **offers a vaccine** (ITT), but not everyone actually takes it.
- The **true treatment effect** depends on those who **receive** the vaccine, which differs from the effect measured under ITT.

Since non-compliance is common in real-world settings, ITT effects are often smaller than true treatment effects. In this case, the difference in observed means between the treatment and control groups is not [Average Treatment Effects], but Intention-to-Treat Effect.

21.10.4 Local Average Treatment Effects

In many empirical settings, not all individuals assigned to treatment actually receive it (**non-compliance**). Instead of estimating the treatment effect for everyone assigned to treatment (i.e., Intention-to-Treat Effects), we often want to estimate the effect of treatment on those who actually **comply** with their assignment.

This is known as the Local Average Treatment Effect, also referred to as the **Complier Average Causal Effect (CACE)**.

- LATE is the treatment effect for the **subgroup of compliers**—those who take the treatment **if and only if** assigned to it.
- Unlike Conditional Average Treatment Effects, which describes heterogeneity across observable subgroups, **LATE focuses on compliance behavior**.
- We typically recover LATE using Instrumental Variables, leveraging random treatment assignment as an instrument.

21.10.4.1 Estimating LATE Using Instrumental Variables

Instrumental variable estimation allows us to isolate the effect of treatment on **compliers** by using **random treatment assignment as an instrument for actual treatment receipt**.

From an instrumental variables perspective, LATE is estimated as:

$$LATE = \frac{ITT}{\text{Share of Compliers}}$$

where:

- **ITT (Intention-to-Treat Effect)** is the effect of being **assigned** to treatment.
- **Share of Compliers** is the proportion of individuals who **actually take** the treatment when assigned to it.

21.10.4.2 Key Properties of LATE

- As the proportion of compliers increases, LATE converges to ITT.
- LATE is always larger than ITT, since ITT averages over both compliers and non-compliers.
- Standard error rule of thumb:
  - The standard error of LATE is given by:

$$SE(LATE) = \frac{SE(ITT)}{\text{Share of Compliers}}$$

- LATE can also be estimated using a pure placebo group (Gerber et al., 2010).

- Partial compliance is difficult to study
- The IV/2SLS estimator is biased in small samples, requiring Bayesian methods for correction (Long et al., 2010; Jin and Rubin, 2009, 2008).

21.10.4.3 One-Sided Noncompliance

One-sided noncompliance occurs when we observe **only compliers and never-takers** in the sample (i.e., no always-takers).

Key assumptions:

- **Exclusion Restriction (Excludability)**: Never-takers have the **same outcomes** regardless of assignment (i.e., treatment has no effect on them because they never receive it).
- **Random Assignment Ensures Balance**: The number of never-takers is expected to be **equal** in the treatment and control groups.

Estimation of LATE under one-sided noncompliance:

$$LATE = \frac{ITT}{\text{Share of Compliers}}$$

Since the never-takers do not receive treatment, this simplifies estimation.

21.10.4.4 Two-Sided Noncompliance

Two-sided noncompliance occurs when we observe **compliers, never-takers, and always-takers** in the sample.

Key assumptions:

- **Exclusion Restriction (Excludability)**: Never-takers and always-takers have the same outcome **regardless of treatment assignment**.
- **Monotonicity Assumption** (No Defiers):
  - There are **no defiers**, meaning no individuals systematically **avoid treatment when assigned** to it.
  - This assumption is standard in practical studies.

Estimation of LATE under two-sided noncompliance:

$$LATE = \frac{ITT}{\text{Share of Compliers}}$$

- Since always-takers receive treatment regardless of assignment, their presence does not bias LATE as long as monotonicity holds.

- In practice, monotonicity is often reasonable, as defiers are rare.

Scenario	What it Measures	When to Use It?	Key Assumptions
Intention-to-Treat	Effect of being assigned to treatment	Policy impact with non-compliance	None (preserves randomization)
LATE	Effect on **compliers only**	When we care about **actual treatment effect** rather than assignment	Excludability, Monotonicity (No Defiers)

21.10.5 Population vs. Sample Average Treatment Effects

In experimental and observational studies, we often estimate the **Sample Average Treatment Effect (SATE)** using a finite sample. However, the **Population Average Treatment Effect (PATE)** is the parameter of interest when making broader generalizations.

Key Issue:
SATE **does not necessarily equal** PATE due to **sample selection bias and treatment imbalance**.

See (Imai et al., 2008) for an in-depth discussion on when SATE diverges from PATE.

Consider a **finite population** of size N from which we observe a sample of size n ($N \gg n$). Half of the sample receives treatment, and half is assigned to control.

Define the following indicators:

- **Sampling Indicator**:

$$I_i = \begin{cases} 1, & \text{if unit } i \text{ is in the sample} \\ 0, & \text{otherwise} \end{cases}$$

- **Treatment Assignment Indicator**:

$$T_i = \begin{cases} 1, & \text{if unit } i \text{ is in the treatment group} \\ 0, & \text{if unit } i \text{ is in the control group} \end{cases}$$

- **Potential Outcomes Framework**:

$$Y_i = \begin{cases} Y_i(1), & \text{if } T_i = 1 \text{ (Treated)} \\ Y_i(0), & \text{if } T_i = 0 \text{ (Control)} \end{cases}$$

- **Observed Outcome**:
  Since we can never observe both potential outcomes for the same unit, the observed outcome is:

$$Y_i | I_i = 1 = T_i Y_i(1) + (1 - T_i) Y_i(0)$$

- **True Individual Treatment Effect**:
  The individual-level treatment effect is:

$$TE_i = Y_i(1) - Y_i(0)$$

However, since we observe only one of $Y_i(1)$ or $Y_i(0)$, TE_i is never directly observed.

21.10.5.1 Definitions of SATE and PATE

- **Sample Average Treatment Effect (SATE):**

$$SATE = \frac{1}{n} \sum_{i \in \{I_i = 1\}} TE_i$$

  SATE is the **average treatment effect within the sample**.

- **Population Average Treatment Effect (PATE):**

$$PATE = \frac{1}{N} \sum_{i=1}^{N} TE_i$$

  PATE represents the **true treatment effect across the entire population**.

Since we observe only a subset of the population, SATE **may not equal** PATE.

21.10.5.2 Decomposing Estimation Error

The baseline estimator for SATE and PATE is the **difference in observed means**:

$$
\begin{aligned}
D &= \frac{1}{n/2} \sum_{i \in (I_i=1, T_i=1)} Y_i - \frac{1}{n/2} \sum_{i \in (I_i=1, T_i=0)} Y_i \\
&= (\text{Mean of Treated Group}) - (\text{Mean of Control Group})
\end{aligned}
$$

Define Δ as the estimation error (i.e., deviation from the truth), under an additive model:

$$
Y_i(t) = g_t(X_i) + h_t(U_i)
$$

The estimation error is decomposed into

$$
\begin{aligned}
PATE - D = \Delta &= \Delta_S + \Delta_T \\
&= (PATE - SATE) + (SATE - D) \\
&= \text{Sample Selection Bias} + \text{Treatment Imbalance} \\
&= (\Delta_{S_X} + \Delta_{S_U}) + (\Delta_{T_X} + \Delta_{T_U}) \\
&= (\text{Selection on Observables} + \text{Selection on Unobservables}) \\
&\quad + (\text{Treatment Imbalance in Observables} + \text{Treatment Imbalance in Unobservables})
\end{aligned}
$$

To further illustrate this, we begin by explicitly defining how the total discrepancy $PATE - D$ separates into different components.

Step 1: From $PATE - D$ to $\Delta_S + \Delta_T$

$$
\underbrace{PATE - D}_{\Delta} = \underbrace{(PATE - SATE)}_{\Delta_S} + \underbrace{(SATE - D)}_{\Delta_T}.
$$

- $PATE - D$: The total discrepancy between the true population treatment effect and the estimate D.
- $\Delta_S = PATE - SATE$: Sample Selection Bias – how much the sample ATE differs from the population ATE.
- $\Delta_T = SATE - D$: Treatment Imbalance – how much the estimated treatment effect deviates from the sample ATE.

Step 2: Breaking Bias into Observables and Unobservables

Each bias term can be decomposed into observed (X) and unobserved (U) factors:

$$
\Delta_S = \underbrace{\Delta_{S_X}}_{\text{Selection on Observables}} + \underbrace{\Delta_{S_U}}_{\text{Selection on Unobservables}}
$$

$$\Delta_T = \underbrace{\Delta_{T_X}}_{\text{Treatment Imbalance in Observables}} + \underbrace{\Delta_{T_U}}_{\text{Treatment Imbalance in Unobservables}}$$

Thus, the final expression:

$$\begin{aligned} PATE - D &= \underbrace{(PATE - SATE)}_{\Delta_S\text{: Sample Selection Bias}} + \underbrace{(SATE - D)}_{\Delta_T\text{: Treatment Imbalance}} \\ &= \underbrace{(\Delta_{S_X} + \Delta_{S_U})}_{\text{Selection on } X + \text{ Selection on } U} + \underbrace{(\Delta_{T_X} + \Delta_{T_U})}_{\text{Imbalance in } X + \text{ Imbalance in } U} . \end{aligned}$$

This decomposition clarifies the sources of error in estimating the true effect, distinguishing between **sample representativeness** (selection bias) and **treatment assignment differences** (treatment imbalance), and further separating these into **observable** and **unobservable** components.

21.10.5.2.1 Sample Selection Bias (Δ_S) Also called **sample selection error**, this arises when the sample **is not representative of the population**.

$$\Delta_S = PATE - SATE = \frac{N-n}{N}(NATE - SATE)$$

where:

- NATE (Non-Sample Average Treatment Effect) is the average treatment effect for the part of the population not included in the sample:

$$NATE = \sum_{i \in (I_i = 0)} \frac{TE_i}{N-n}$$

To eliminate sample selection bias ($\Delta_S = 0$):

1. Redefine the sample as the entire population ($N = n$).
2. Ensure $NATE = SATE$ (e.g., treatment effects must be homogeneous across sampled and non-sampled units).

However, when treatment effects vary across individuals, random sampling only warrants sample selection bias but does not sample eliminate error.

21.10.5.2.2 Treatment Imbalance Error (Δ_T) Also called **treatment imbalance bias**, this occurs when **the empirical distribution of treated and control units differs**.

$$\Delta_T = SATE - D$$

Key insight:
$\Delta_T \to 0$ when the treatment and control groups are balanced across both observables (X) and unobservables (U).

Since we cannot directly adjust for unobservables, imbalance correction methods focus on observables.

21.10.5.3 Adjusting for (Observable) Treatment Imbalance

However, in real-world studies:

- We can only adjust for observables X, not unobservables U.
- Residual imbalance in unobservables may still introduce bias after adjustment.

To address treatment imbalance, researchers commonly use:

1. Blocking
2. Matching Methods

Method	Blocking	Matching Methods
Definition	Random assignment within predefined strata based on pre-treatment covariates.	Dropping, repeating, or grouping observations to balance covariates between treated and control groups (Rubin, 1973).
When Applied?	Before treatment assignment (in experimental designs).	After treatment assignment (in observational studies).
Effectiveness	Ensures exact balance within strata but may require large sample sizes for fine stratification.	Can improve balance, but risk of increasing bias if covariates are poorly chosen.
What If Covariates Are Irrelevant?	No effect on treatment estimates.	Worst-case scenario: If matching is done on covariates uncorrelated with treatment but correlated with outcomes, it may increase bias instead of reducing it.

Method	Blocking	Matching Methods
Benefits	Eliminates imbalance in observables ($\Delta_{T_X} = 0$). Effect on unobservables is uncertain (may help if unobservables correlate with observables).	Reduces model dependence, bias, variance, and mean-squared error (MSE). Matching only balances observables, and its effect on unobservables is unknown.

21.10.6 Average Treatment Effects on the Treated and Control

In many empirical studies, researchers are interested in how treatment affects **specific subpopulations** rather than the entire population. Two commonly used treatment effect measures are:

1. **Average Treatment Effect on the Treated** (ATT): The effect of treatment on individuals who actually received treatment.
2. **Average Treatment Effect on the Control** (ATC): The effect treatment would have had on individuals who were not treated.

Understanding the distinction between ATT, ATC, and ATE is crucial for determining external validity and for designing targeted policies.

21.10.6.1 Average Treatment Effect on the Treated

The **ATT** measures the expected treatment effect **only for those who were actually treated**:

$$\begin{aligned} ATT &= E[Y_i(1) - Y_i(0)|D_i = 1] \\ &= E[Y_i(1)|D_i = 1] - E[Y_i(0)|D_i = 1] \end{aligned}$$

Key Interpretation:

- ATT tells us how much better (or worse) off treated individuals are compared to their hypothetical counterfactual outcome (had they not been treated).
- It is useful for evaluating the effectiveness of interventions on those who self-select into treatment.

21.10.6.2 Average Treatment Effect on the Control

The **ATC** measures the expected treatment effect **only for those who were not treated**:

$$\begin{aligned} ATC &= E[Y_i(1) - Y_i(0)|D_i = 0] \\ &= E[Y_i(1)|D_i = 0] - E[Y_i(0)|D_i = 0] \end{aligned}$$

Key Interpretation:

- ATC answers the question: "What would have been the effect of treatment if it had been given to those who were not treated?"
- It is important for understanding how an intervention might generalize to untreated populations.

21.10.6.3 Relationship Between ATT, ATC, and ATE

Under **random assignment and full compliance**, we have:

$$ATE = ATT = ATC$$

Why?

- Randomization ensures that treated and untreated groups are statistically identical before treatment.
- Thus, treatment effects are the same across groups, leading to ATT = ATC = ATE.

However, in observational settings, selection bias and treatment heterogeneity may cause ATT and ATC to diverge from ATE.

21.10.6.4 Sample Average Treatment Effect on the Treated

The **Sample ATT (SATT)** is the empirical estimate of ATT in a finite sample:

$$SATT = \frac{1}{n} \sum_{i \in D_i = 1} TE_i$$

where:

- $TE_i = Y_i(1) - Y_i(0)$ is the treatment effect for unit i.
- n is the number of treated units in the sample.
- The summation is taken only over treated units in the sample.

21.10.6.5 Population Average Treatment Effect on the Treated

The **Population ATT (PATT)** generalizes ATT to the entire treated population:

$$PATT = \frac{1}{N} \sum_{i \in D_i = 1} TE_i$$

where:

- $TE_i = Y_i(1) - Y_i(0)$ is the treatment effect for unit i.
- N is the total number of treated units in the population.
- The summation is taken over all treated individuals in the population.

If the sample is **randomly drawn**, then $SATT \approx PATT$, but if the sample is **not representative**, $SATT$ may **overestimate or underestimate** $PATT$.

21.10.6.6 When ATT and ATC Diverge from ATE

In real-world studies, ATT and ATC often **differ from ATE** due to **treatment effect heterogeneity** and **selection bias**.

21.10.6.6.1 Selection Bias in ATT If individuals **self-select** into treatment, then the treated group may be **systematically different** from the control group.

- **Example**:
  - Suppose a job training program is voluntary.
  - Individuals who enroll might be more motivated or have better skills than those who do not.
  - As a result, the treatment effect (ATT) may not generalize to the untreated group (ATC).

This implies:

$$ATT \neq ATC$$

unless treatment assignment is random.

21.10.6.6.2 Treatment Effect Heterogeneity If **treatment effects vary** across individuals, then:

- ATT may be larger or smaller than ATE, depending on how treatment effects differ across subgroups.

- ATC may be larger or smaller than ATT, if the untreated group would have responded differently to treatment.

Example:

- A scholarship program may be more beneficial for students from lower-income families than for students from wealthier backgrounds.
- If lower-income students are more likely to apply for the scholarship, then ATT > ATE.
- However, if wealthier students (who did not receive the scholarship) would have benefited less from it, then ATC < ATE.

Thus, we may observe:

$$ATE \neq ATT \neq ATC$$

Treatment Effect	**Definition**	**Use Case**	**Potential Issues**
ATE (Average Treatment Effect)	Effect on **randomly selected individuals**	Policy decisions applicable to entire population	Requires full randomization
ATT (Average Treatment on Treated)	Effect on **those who received treatment**	Evaluating effectiveness of interventions for targeted groups	Selection bias if treatment is voluntary
ATC (Average Treatment on Control)	Effect **if treatment were given to untreated individuals**	Predicting treatment effects for new populations	May not be generalizable

21.10.7 Quantile Average Treatment Effects

Instead of focusing on the mean effect (ATE), Quantile Treatment Effects (QTE) help us understand how treatment shifts the entire distribution of an outcome variable.

The Quantile Treatment Effect at quantile τ is defined as:

$$QTE_\tau = Q_\tau(Y_1) - Q_\tau(Y_0)$$

where:

- $Q_\tau(Y_1)$ is the τ-th quantile of the outcome distribution **under treatment**.

- $Q_\tau(Y_0)$ is the τ-th quantile of the outcome distribution **under control**.

When to Use QTE?

- **Heterogeneous Treatment Effects**: If treatment effects differ across individuals, ATE may be misleading.
- **Policy Targeting**: Policymakers may care more about low-income individuals (e.g., bottom 25%) rather than the average effect.
- **Distributional Changes**: QTE allows us to assess whether treatment increases inequality (e.g., benefits the rich more than the poor).

Estimation of QTE

QTE can be estimated using:

- **Quantile Regression**: Extends linear regression to estimate effects at different quantiles.
- **Instrumental Variables for QTE**: Requires additional assumptions to estimate causal effects in the presence of endogeneity (Abadie et al., 2002; Chernozhukov and Hansen, 2005).

Example: Wage Policy Impact

- Suppose a minimum wage increase is introduced.
- The ATE might show a small positive effect on earnings.
- However, QTE might reveal:
  - No effect at the bottom quantiles (for workers who lose jobs).
  - A positive effect at the median.
  - A strong positive effect at the top quantiles (for experienced workers who benefit the most).

Thus, QTE provides a more detailed view of the treatment effect across the entire income distribution.

21.10.8 Log-Odds Treatment Effects for Binary Outcomes

When the **outcome variable is binary** (e.g., success/failure, employed/unemployed, survived/died), it is often useful to measure the treatment effect in **log-odds form**.

For a binary outcome Y, define the **probability of success** as:

$$P(Y = 1|D = d)$$

The **log-odds of success** under treatment and control are:

$$\text{Log-odds}(Y|D = 1) = \log\left(\frac{P(Y = 1|D = 1)}{1 - P(Y = 1|D = 1)}\right)$$

$$\text{Log-odds}(Y|D=0) = \log\left(\frac{P(Y=1|D=0)}{1-P(Y=1|D=0)}\right)$$

The **Log-Odds Treatment Effect (LOTE)** is then:

$$LOTE = \text{Log-odds}(Y|D=1) - \text{Log-odds}(Y|D=0)$$

This captures how treatment **affects the relative likelihood** of success in a nonlinear way.

When to Use Log-Odds Treatment Effects?

- **Binary Outcomes**: When the treatment outcome is 0 or 1 (e.g., employed/unemployed).
- **Nonlinear Treatment Effects**: Log-odds help handle situations where effects are multiplicative rather than additive.
- **Rare Events**: Useful in cases where the outcome probability is very small or very large.

Estimation of Log-Odds Treatment Effects

- **Logistic Regression with Treatment Indicator**:

  $$\log\left(\frac{P(Y=1|D=1)}{1-P(Y=1|D=1)}\right) = \beta_0 + \beta_1 D$$

  where β_1 represents the **log-odds treatment effect**.

- **Randomization-Based Estimation**: Freedman (2008) provides a framework for randomized trials that ensures consistent estimation.

- **Attributable Effects**: Alternative methods, such as those in (Rosenbaum, 2002), estimate the proportion of cases attributable to the treatment.

21.10.9 Summary Table: Treatment Effect Estimands

Treatment Effect	**Definition**	**Use Case**	**Key Assumptions**	**When It Differs from ATE?**
Average Treatment Effect	The expected treatment effect for a randomly chosen individual in the population.	General policy evaluation; measures the overall impact.	Randomization or strong ignorability (treatment assignment independent of potential outcomes).	-
Conditional Average Treatment Effect	The treatment effect for a specific subgroup of the population, conditional on covariates X.	Identifies heterogeneous effects; useful for targeted interventions.	Treatment effect heterogeneity must exist.	Differs when treatment effects vary across subgroups.
Intention-to-Treat Effect	The effect of being assigned to treatment, regardless of actual compliance.	Policy evaluations where non-compliance exists.	Randomized treatment assignment ensures unbiased estimation.	Lower than ATE when not all assigned individuals comply.
Local Average Treatment Effect	The effect of treatment only on compliers—those who take the treatment if and only if assigned to it.	When compliance is imperfect, LATE isolates the effect for compliers.	Monotonicity (no defiers); instrument only affects the outcome through treatment.	Differs from ATE when compliance is selective.
Average Treatment Effect on the Treated	The effect of treatment on those who actually received the treatment.	Used when assessing effectiveness of a treatment for those who self-select into it.	No unmeasured confounders within the treated group.	Differs when treatment selection is not random.

Treatment Effect	**Definition**	**Use Case**	**Key Assumptions**	**When It Differs from ATE?**
Average Treatment Effect on the Control	The effect the treatment would have had on individuals who were not treated.	Predicts the effect of expanding a program to the untreated population.	No unmeasured confounders within the control group.	Differs when treatment effects are heterogeneous.
Sample Average Treatment Effect	The estimated treatment effect in the sample.	Used when evaluating treatment within a specific sample.	Sample must be representative of the population for external validity.	Differs when the sample is not representative of the population.
Population Average Treatment Effect	The expected treatment effect for the entire population.	Policy design and large-scale decision-making.	Requires that sample selection is random.	Differs when sample selection bias exists.
Quantile Treatment Effect	The treatment effect at a specific percentile of the outcome distribution.	Understanding distributional effects rather than mean effects.	Rank preservation or monotonicity assumptions may be needed.	Differs when treatment effects vary across outcome quantiles.
Log-Odds Treatment Effect	The effect of treatment on binary outcomes, expressed in log-odds.	Used when outcomes are dichotomous (e.g., employed/unemployed, survived/died).	Logistic model assumptions must hold.	Differs when treatment effects are nonlinear or outcome probabilities are low.

A. EXPERIMENTAL DESIGN

Chapter 22

Experimental Design

Imagine you're a marketing manager trying to decide whether a new advertising campaign will boost sales. Or perhaps you're an economist investigating the impact of tax incentives on consumer spending. In both cases, you need a way to determine **causal effects**—not just correlations. This is where **experimental design** becomes essential.

At its core, experimental design ensures that studies are conducted efficiently and that valid conclusions can be drawn. In business applications, experiments help quantify the effects of pricing strategies, marketing tactics, and financial interventions. A well-designed experiment **reduces bias**, **controls variability**, and **maximizes the accuracy of causal inferences**.

22.1 Principles of Experimental Design

A well-designed experiment follows three key principles:

1. **Randomization**: Ensures that subjects or experimental units are assigned to different groups randomly, eliminating selection bias.
2. **Replication**: Increases the precision of estimates by repeating the experiment on multiple subjects.
3. **Control**: Isolates the effect of treatments by using control groups or baseline conditions to minimize confounding factors.

In addition to these, **blocking** and **factorial designs** further refine experimental accuracy.

22.2 The Gold Standard: Randomized Controlled Trials

Randomized Controlled Trials (RCTs) are the **holy grail of causal inference**. Their power comes from **random assignment**, which ensures that any differences between treatment and control groups arise **only** due to the intervention.

RCTs provide:

- **Unbiased estimates** of treatment effects
- **Elimination of confounding factors on average** (although covariate imbalance can occur, necessitating techniques like [Rerandomization] to achieve a "platinum standard" set by Tukey (1993))

An RCT consists of two groups:

1. **Treatment group**: Receives the intervention (e.g., a new marketing campaign, drug, or financial incentive).
2. **Control group**: Does not receive the intervention, serving as a baseline.

Subjects from the same population are **randomly assigned** to either group. This randomization ensures that **any observed differences in outcomes are due solely to the treatment**—not external factors.

However, RCTs are easier to conduct in **hard sciences** (e.g., medicine or physics), where treatments and environments can be tightly controlled. In **social sciences**, challenges arise because:

- Human behavior is unpredictable.
- Some treatments are impossible or unethical to introduce (e.g., assigning individuals to poverty).
- Real-world environments are difficult to control.

To address these challenges, social scientists use Quasi-Experimental Methods to approximate experimental conditions.

RCTs establish internal validity, meaning that the observed treatment effects are **causally interpretable**. Even though **random assignment is not the same as *ceteris paribus*** (holding everything else constant), it achieves a similar effect: **on average**, treatment and control groups should be comparable.

22.3 Selection Problem

A fundamental challenge in causal inference is that we never observe both potential outcomes for the same individual—only one or the other. This creates the selection problem, which we formalize below.

Assume we have:

- A **binary treatment variable**:
  $D_i \in \{0, 1\}$, where:
  - $D_i = 1$ indicates that individual i receives the treatment.
  - $D_i = 0$ indicates that individual i does **not** receive the treatment.
- The **outcome of interest**:
  Y_i, which depends on whether the individual is treated or not:
  - Y_{0i}: The outcome **if not treated**.
  - Y_{1i}: The outcome **if treated**.

Thus, the **potential outcomes framework** is defined as:

$$\text{Potential Outcome} = \begin{cases} Y_{1i}, & \text{if } D_i = 1 \quad \text{(Treated)} \\ Y_{0i}, & \text{if } D_i = 0 \quad \text{(Untreated)} \end{cases}$$

However, we only **observe** one outcome per individual:

$$Y_i = Y_{0i} + (Y_{1i} - Y_{0i})D_i$$

This means that for any given person, we either observe Y_{1i} or Y_{0i}, but **never both**. Since we cannot observe counterfactuals (unless we invent a time machine), we must rely on statistical inference to estimate treatment effects.

22.3.1 The Observed Difference in Outcomes

The goal is to estimate the difference in expected outcomes between treated and untreated individuals:

$$E[Y_i|D_i = 1] - E[Y_i|D_i = 0]$$

Expanding this equation:

$$\begin{aligned} E[Y_i|D_i = 1] - E[Y_i|D_i = 0] &= (E[Y_{1i}|D_i = 1] - E[Y_{0i}|D_i = 1]) + (E[Y_{0i}|D_i = 1] - E[Y_{0i}|D_i = 0]) \\ &= (E[Y_{1i} - Y_{0i}|D_i = 1]) + (E[Y_{0i}|D_i = 1] - E[Y_{0i}|D_i = 0]) \end{aligned}$$

This equation decomposes the **observed difference** into two components:

-
- **Selection Bias**:
  $E[Y_{0i}|D_i = 1] - E[Y_{0i}|D_i = 0]$, which captures systematic differences between treated and untreated groups **even in the absence of treatment**.

Thus, the **observed difference in outcomes** is:

$$\text{Observed Difference} = \text{ATT} + \text{Selection Bias}$$

22.3.2 Eliminating Selection Bias with Random Assignment

With **random assignment** of treatment, D_i is independent of potential outcomes:

$$E[Y_i|D_i = 1] - E[Y_i|D_i = 0] = E[Y_{1i} - Y_{0i}]$$

This works because, under true randomization:

$$E[Y_{0i}|D_i = 1] = E[Y_{0i}|D_i = 0]$$

which eliminates selection bias. Consequently, the observed difference now **directly estimates the true causal effect**:

$$E[Y_i|D_i = 1] - E[Y_i|D_i = 0] = E[Y_{1i} - Y_{0i}]$$

Thus, randomized controlled trials provide an unbiased estimate of the average treatment effect.

22.3.3 Another Representation Under Regression

So far, we have framed the selection problem using expectations and potential outcomes. Another way to represent treatment effects is through **regression models**, which provide a practical framework for estimation.

Suppose the treatment effect is constant across individuals:

$$Y_{1i} - Y_{0i} = \rho$$

This implies that each treated individual experiences the same treatment effect (ρ), though their baseline outcomes (Y_{0i}) may vary.

Since we only observe **one** of the potential outcomes, the observed outcome can be expressed as:

$$\begin{aligned} Y_i &= E(Y_{0i}) + (Y_{1i} - Y_{0i})D_i + [Y_{0i} - E(Y_{0i})] \\ &= \alpha + \rho D_i + \eta_i \end{aligned}$$

where:

- $\alpha = E(Y_{0i})$, the expected outcome for untreated individuals.
- ρ represents the **causal treatment effect**.

- $\eta_i = Y_{0i} - E(Y_{0i})$, capturing individual deviations from the mean untreated outcome.

Thus, the regression model provides an intuitive way to express treatment effects.

22.3.3.1 Conditional Expectations and Selection Bias

Taking expectations conditional on treatment status:

$$E[Y_i|D_i = 1] = \alpha + \rho + E[\eta_i|D_i = 1]$$
$$E[Y_i|D_i = 0] = \alpha + E[\eta_i|D_i = 0]$$

The observed difference in means between treated and untreated groups is:

$$E[Y_i|D_i = 1] - E[Y_i|D_i = 0] = \rho + E[\eta_i|D_i = 1] - E[\eta_i|D_i = 0]$$

Here, the term $E[\eta_i|D_i = 1] - E[\eta_i|D_i = 0]$ represents **selection bias**—the correlation between the regression error term (η_i) and the treatment variable (D_i).

Under random assignment, we assume that potential outcomes are **independent** of treatment (D_i):

$$E[\eta_i|D_i = 1] - E[\eta_i|D_i = 0] = E[Y_{0i}|D_i = 1] - E[Y_{0i}|D_i = 0] = 0$$

Thus, under true **randomization**, selection bias disappears, and the observed difference directly estimates the **causal effect** ρ.

22.3.3.2 Controlling for Additional Variables

In many real-world scenarios, **random assignment is imperfect**, and selection bias may still exist. To mitigate this, we introduce **control variables** (X_i), such as demographic characteristics, firm size, or prior purchasing behavior.

If X_i is **uncorrelated with treatment** (D_i), including it in our regression model does not bias the estimate of ρ but has two advantages:

1. **It reduces residual variance** (η_i), improving the precision of ρ.
2. **It accounts for additional sources of variability**, making the model more robust.

Thus, our regression model extends to:

$$Y_i = \alpha + \rho D_i + X_i'\gamma + \eta_i$$

where:

- X_i represents a vector of control variables.
- γ captures the effect of X_i on the outcome.

22.3.3.3 Example: Racial Discrimination in Hiring

A famous study by Bertrand and Mullainathan (2004) examined racial discrimination in hiring by randomly assigning Black- and White-sounding names to identical job applications. By ensuring that names were assigned randomly, the authors eliminated confounding factors like education and experience, allowing them to estimate the causal effect of race on callback rates.

22.4 Classical Experimental Designs

Experimental designs provide structured frameworks for conducting experiments, ensuring that results are **statistically valid** and **practically applicable**. The choice of design depends on the research question, the nature of the treatment, and potential sources of variability. For a more in-depth statistical understanding of these designs, we will revisit them again in Analysis of Variance.

22.4.1 Completely Randomized Design

In a Completely Randomized Design (CRD), each experimental unit is randomly assigned to a treatment group. This is the simplest form of experimental design and is effective when no confounding factors are present.

Example: Email Marketing Experiment

A company tests three different email marketing strategies (A, B, and C) to measure their effect on **customer engagement** (click-through rate). Customers are randomly assigned to receive one of the three emails.

Mathematical Model

$$Y_{ij} = \mu + \tau_i + \epsilon_{ij}$$

where:

- Y_{ij} is the response variable (e.g., click-through rate).
- μ is the overall mean response.
- τ_i is the effect of treatment i.
- ϵ_{ij} is the random error term, assumed to be normally distributed: $\epsilon_{ij} \sim N(0, \sigma^2)$.

```r
set.seed(123)

# Simulated dataset for email marketing experiment
data <- data.frame(
  group = rep(c("A", "B", "C"), each = 10),
  response = c(rnorm(10, mean=50, sd=5),
               rnorm(10, mean=55, sd=5),
               rnorm(10, mean=60, sd=5))
)

# ANOVA to test for differences among groups
anova_result <- aov(response ~ group, data = data)
summary(anova_result)
#>             Df Sum Sq Mean Sq F value  Pr(>F)
#> group        2  306.1  153.04   6.435 0.00518 **
#> Residuals   27  642.1   23.78
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If the **p-value** in the ANOVA summary is **less than 0.05**, we reject the null hypothesis and conclude that at least one email strategy significantly affects engagement.

22.4.2 Randomized Block Design

A Randomized Block Design (RBD) is used when experimental units can be grouped into homogeneous blocks based on a known confounding factor. Blocking helps reduce unwanted variation, increasing the precision of estimated treatment effects.

Example: Store Layout Experiment

A retailer tests three store layouts (A, B, and C) on sales performance. Since store location (Urban, Suburban, Rural) might influence sales, we use blocking to control for this effect.

Mathematical Model

$$Y_{ij} = \mu + \tau_i + \beta_j + \epsilon_{ij}$$

where:

- Y_{ij} is the sales outcome for store i in location j.
- μ is the overall mean sales.
- τ_i is the effect of layout i.
- β_j represents the block effect (location).

- ϵ_{ij} is the random error.

```
set.seed(123)

# Simulated dataset for store layout experiment
data <- data.frame(
  block = rep(c("Urban", "Suburban", "Rural"), each = 6),
  layout = rep(c("A", "B", "C"), times = 6),
  sales = c(rnorm(6, mean=200, sd=20),
            rnorm(6, mean=220, sd=20),
            rnorm(6, mean=210, sd=20))
)

# ANOVA with blocking factor
anova_block <- aov(sales ~ layout + block, data = data)
summary(anova_block)
#>             Df Sum Sq Mean Sq F value Pr(>F)
#> layout       2     71    35.7   0.071  0.931
#> block        2    328   164.1   0.328  0.726
#> Residuals   13   6500   500.0
```

By including `block` in the model, we account for **location effects**, leading to **more accurate** treatment comparisons.

22.4.3 Factorial Design

A Factorial Design evaluates the effects of two or more factors simultaneously, allowing for the study of interactions between variables.

Example: Pricing and Advertising Experiment

A company tests two pricing strategies (High, Low) and two advertising methods (TV, Social Media) on sales.

Mathematical Model

$$Y_{ijk} = \mu + \tau_i + \gamma_j + (\tau\gamma)_{ij} + \epsilon_{ijk}$$

where:

- τ_i is the effect of price level i.
- γ_j is the effect of advertising method j.
- $(\tau\gamma)_{ij}$ is the **interaction effect** between price and advertising.
- ϵ_{ijk} is the random error term.

```
set.seed(123)

# Simulated dataset
```

```r
data <- expand.grid(
  Price = c("High", "Low"),
  Advertising = c("TV", "Social Media"),
  Replicate = 1:10
)

# Generate response variable (sales)
data$Sales <- with(data,
                   100 +
                   ifelse(Price == "Low", 10, 0) +
                   ifelse(Advertising == "Social Media", 15, 0) +
                   ifelse(Price == "Low" & Advertising == "Social Media", 5, 0) +
                   rnorm(nrow(data), sd=5))

# Two-way ANOVA
anova_factorial <- aov(Sales ~ Price * Advertising, data = data)
summary(anova_factorial)
#>                    Df Sum Sq Mean Sq F value   Pr(>F)
#> Price               1   1364    1364   66.60 1.05e-09 ***
#> Advertising         1   3640    3640  177.67 1.72e-15 ***
#> Price:Advertising   1     15      15    0.71    0.405
#> Residuals          36    738      20
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

22.4.4 Crossover Design

A Crossover Design is used when each subject receives multiple treatments in a sequential manner. This design controls for individual differences by using each subject as their own control.

Example: Drug Trial

Patients receive Drug A in the first period and Drug B in the second period, or vice versa.

Mathematical Model

$$Y_{ijk} = \mu + \tau_i + \pi_j + \beta_k + \epsilon_{ijk}$$

where:

- τ_i is the **treatment effect**.
- π_j is the **period effect** (e.g., learning effects).
- β_k is the **subject effect** (individual baseline differences).

```r
set.seed(123)

# Simulated dataset
data <- data.frame(
  Subject = rep(1:10, each = 2),
  Period = rep(c("Period 1", "Period 2"), times = 10),
  Treatment = rep(c("A", "B"), each = 10),
  Response = c(rnorm(10, mean=50, sd=5), rnorm(10, mean=55, sd=5))
)

# Crossover ANOVA
anova_crossover <- aov(Response ~ Treatment + Period + Error(Subject/Period), data = da
summary(anova_crossover)
#>
#> Error: Subject
#>           Df Sum Sq Mean Sq
#> Treatment  1  63.94   63.94
#>
#> Error: Subject:Period
#>        Df Sum Sq Mean Sq
#> Period  1  21.92   21.92
#>
#> Error: Within
#>           Df Sum Sq Mean Sq F value Pr(>F)
#> Treatment  1  134.9  134.91   5.231 0.0371 *
#> Period     1    0.2    0.24   0.009 0.9237
#> Residuals 15  386.9   25.79
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

22.4.5 Split-Plot Design

A Split-Plot Design is used when one factor is applied at the group (whole-plot) level and another at the individual (sub-plot) level. This design is particularly useful when some factors are harder or more expensive to randomize than others.

Example: Farming Experiment

A farm is testing two irrigation methods (Drip vs. Sprinkler) and two soil types (Clay vs. Sand) on crop yield. Since irrigation systems are installed at the farm level and are difficult to change, they are treated as the whole-plot factor. However, different soil types exist within each farm and can be tested more easily, making them the sub-plot factor.

Mathematical Model

The statistical model for a **Split-Plot Design** is:

$$Y_{ijk} = \mu + \alpha_i + B_k + \beta_j + (\alpha\beta)_{ij} + \epsilon_{ijk}$$

where:

- Y_{ijk} is the response (e.g., crop yield).
- μ is the overall mean.
- α_i is the **whole-plot factor** (Irrigation method).
- B_k is the **random block effect** (Farm-level variation).
- β_j is the **sub-plot factor** (Soil type).
- $(\alpha\beta)_{ij}$ is the interaction effect between **Irrigation** and **Soil type**.
- $\epsilon_{ijk} \sim N(0, \sigma^2)$ represents the **random error term**.

The key feature of the Split-Plot Design is that the whole-plot factor (α_i) is tested against the farm-level variance (B_k), while the sub-plot factor (β_j) is tested against individual variance (ϵ_{ijk}).

We model the **Split-Plot Design** using a **Mixed Effects Model**, treating **Farm** as a **random effect** to account for variation at the whole-plot level.

```r
set.seed(123)

# Simulated dataset for a split-plot experiment
data <- data.frame(
  Farm = rep(1:6, each = 4), # 6 farms (whole plots)
  Irrigation = rep(c("Drip", "Sprinkler"), each = 12), # Whole-plot factor
  Soil = rep(c("Clay", "Sand"), times = 12), # Sub-plot factor
  Yield = c(rnorm(12, mean=30, sd=5), rnorm(12, mean=35, sd=5)) # Response variable
)

# Load mixed-effects model library
library(lme4)

# Mixed-effects model: Whole-plot factor (Irrigation) as a random effect
model_split <- lmer(Yield ~ Irrigation * Soil + (1 | Farm), data = data)

# Summary of the model
summary(model_split)
#> Linear mixed model fit by REML ['lmerMod']
```

```
#> Formula: Yield ~ Irrigation * Soil + (1 | Farm)
#>    Data: data
#>
#> REML criterion at convergence: 128.1
#>
#> Scaled residuals:
#>      Min       1Q   Median       3Q      Max
#> -1.72562 -0.57572 -0.09767  0.60248  2.04346
#>
#> Random effects:
#>  Groups   Name        Variance Std.Dev.
#>  Farm     (Intercept)  0.00    0.000
#>  Residual             24.79    4.979
#> Number of obs: 24, groups:  Farm, 6
#>
#> Fixed effects:
#>                              Estimate Std. Error t value
#> (Intercept)                    31.771      2.033  15.629
#> IrrigationSprinkler             2.354      2.875   0.819
#> SoilSand                       -1.601      2.875  -0.557
#> IrrigationSprinkler:SoilSand    1.235      4.066   0.304
#>
#> Correlation of Fixed Effects:
#>             (Intr) IrrgtS SolSnd
#> IrrgtnSprnk -0.707
#> SoilSand    -0.707  0.500
#> IrrgtnSp:SS  0.500 -0.707 -0.707
#> optimizer (nloptwrap) convergence code: 0 (OK)
#> boundary (singular) fit: see help('isSingular')
```

In this model:

- Irrigation (Whole-plot factor) is tested against Farm-level variance.
- Soil type (Sub-plot factor) is tested against Residual variance.
- The interaction between Irrigation × Soil is also evaluated.

This hierarchical structure accounts for the fact that farms are not independent, improving the precision of our estimates.

22.4.6 Latin Square Design

When two potential confounding factors exist, Latin Square Designs provide a structured way to control for these variables. This design is common in scheduling, manufacturing, and supply chain experiments.

Example: Assembly Line Experiment

A manufacturer wants to test three assembly methods (A, B, C) while controlling for work shifts and workstations. Since both shifts and workstations may influence production time, a Latin Square Design ensures that each method is tested once per shift and once per workstation.

Mathematical Model

A Latin Square Design ensures that each treatment appears exactly once in each row and column:

$$Y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_k + \epsilon_{ijk}$$

where:

- Y_{ijk} is the outcome (e.g., assembly time).
- μ is the overall mean.
- α_i is the **treatment effect** (Assembly Method).
- β_j is the **row effect** (Work Shift).
- γ_k is the **column effect** (Workstation).
- ϵ_{ijk} is the random error term.

This ensures that each treatment is **equally balanced across both confounding factors**.

We implement a **Latin Square Design** by treating **Assembly Method** as the primary factor, while controlling for **Shifts** and **Workstations**.

```r
set.seed(123)

# Define the Latin Square layout
latin_square <- data.frame(
  Shift = rep(1:3, each = 3), # Rows
  Workstation = rep(1:3, times = 3), # Columns
  Method = c("A", "B", "C", "C", "A", "B", "B", "C", "A"), # Treatments assigned in a balanced wa
  Time = c(rnorm(3, mean=30, sd=3), rnorm(3, mean=28, sd=3), rnorm(3, mean=32, sd=3)) # Assembly
)

# ANOVA for Latin Square Design
anova_latin <- aov(Time ~ factor(Shift) + factor(Workstation) + factor(Method), data = latin_squa
summary(anova_latin)
#>                     Df Sum Sq Mean Sq F value Pr(>F)
#> factor(Shift)        2  1.148   0.574   0.079  0.927
#> factor(Workstation)  2 24.256  12.128   1.659  0.376
#> factor(Method)       2 14.086   7.043   0.964  0.509
#> Residuals            2 14.619   7.310
```

- If the p-value for "Method" is significant, we conclude that different assembly methods impact production time.
- If "Shift" or "Workstation" is significant, it indicates systematic differences across these variables.

22.5 Advanced Experimental Designs

22.5.1 Semi-Random Experiments

In **semi-random experiments**, participants are **not fully randomized** into treatment and control groups. Instead, a **structured randomization process** ensures fairness while allowing some level of causal inference.

22.5.1.1 Example: Loan Assignment Fairness

A bank wants to evaluate a new loan approval policy while ensuring that the experiment does not unfairly exclude specific demographics.

To maintain fairness:

1. Applicants are first stratified based on income and credit history.
2. Within each stratum, a random subset is assigned to receive the new policy, while others continue under the old policy.

```
set.seed(123)

# Create stratified groups
data <- data.frame(
  income_group = rep(c("Low", "Medium", "High"), each = 10),
  treatment = sample(rep(c("New Policy", "Old Policy"), each = 15)) # Stratified rando
)

# Display the stratification results
table(data$income_group, data$treatment)
#>
#>          New Policy Old Policy
#>   High            6          4
#>   Low             6          4
#>   Medium          3          7
```

This approach ensures that each income group is fairly represented in both treatment and control conditions.

22.5.1.2 Case Study: Chicago Open Enrollment Program

A well-known example of semi-random assignment is the **Chicago Open Enrollment Program** (Cullen et al., 2005), where students could apply to choice

schools.

However, since many schools were oversubscribed (i.e., demand exceeded supply), they used a random lottery system to allocate spots.

Thus, while **enrollment itself was not fully random**, the **lottery outcomes were random**, allowing researchers to estimate the Intent-to-Treat effect while acknowledging that not all students who won the lottery actually enrolled.

This situation presents a classic case where:

- **School choice is not random**: Families self-select into applying to certain schools.
- **Lottery outcomes are random**: Among those who apply, winning or losing the lottery is as good as a **random assignment**.

Let:

- $Enroll_{ij} = 1$ if student i enrolls in school j, and 0 otherwise.
- $Win_{ij} = 1$ if student i **wins the lottery**, and 0 otherwise.
- $Apply_{ij} = 1$ if student i **applies** to school j.

We define:

$$\delta_j = E[Y_i|Enroll_{ij} = 1, Apply_{ij} = 1] - E[Y_i|Enroll_{ij} = 0, Apply_{ij} = 1]$$

$$\theta_j = E[Y_i|Win_{ij} = 1, Apply_{ij} = 1] - E[Y_i|Win_{ij} = 0, Apply_{ij} = 1]$$

where:

- δ_j is the **treatment effect** (impact of actual school enrollment).
- θ_j is the intent-to-treat effect (impact of winning the lottery).

Since **not all winners enroll**, we know that:

$$\delta_j \neq \theta_j$$

Thus, we can only estimate θ_j directly, and need an additional method to recover δ_j.

This distinction is crucial because simply comparing lottery winners and losers does not measure the true effect of enrollment—only the effect of being given the opportunity to enroll.

To estimate the treatment effect (δ_j), we use an Instrumental Variable approach:

$$\delta_j = \frac{E[Y_i|W_{ij} = 1, A_{ij} = 1] - E[Y_i|W_{ij} = 0, A_{ij} = 1]}{P(Enroll_{ij} = 1|W_{ij} = 1, A_{ij} = 1) - P(Enroll_{ij} = 1|W_{ij} = 0, A_{ij} = 1)}$$

where:

- $P(Enroll_{ij} = 1|W_{ij} = 1, A_{ij} = 1)$ = probability of enrolling **if winning the lottery**.
- $P(Enroll_{ij} = 1|W_{ij} = 0, A_{ij} = 1)$ = probability of enrolling **if losing the lottery**.

This adjustment accounts for the fact that some lottery winners do not enroll, and thus the observed effect (θ_j) underestimates the true treatment effect (δ_j).

This Instrumental Variable approach corrects for **selection bias** by leveraging **randomized lottery assignment**.

Numerical Example

Assume **10 students win** the lottery and **10 students lose**.

For Winners:

Type	Count	Selection Effect	Treatment Effect	Total Effect
Always Takers	1	+0.2	+1	+1.2
Compliers	2	0	+1	+1
Never Takers	7	-0.1	0	-0.1

For Losers:

Type	Count	Selection Effect	Treatment Effect	Total Effect
Always Takers	1	+0.2	+1	+1.2
Compliers	2	0	0	0
Never Takers	7	-0.1	0	-0.1

Computing Intent-to-Treat Effect

We compute the **expected outcome** for those who **won** and **lost** the lottery.

$$E[Y_i|W_{ij} = 1, A_{ij} = 1] = \frac{1(1.2) + 2(1) + 7(-0.1)}{10} = 0.25$$
$$E[Y_i|W_{ij} = 0, A_{ij} = 1] = \frac{1(1.2) + 2(0) + 7(-0.1)}{10} = 0.05$$

Thus, the Intent-to-Treat Effect is:

$$\text{Intent-to-Treat Effect} = 0.25 - 0.05 = 0.2$$

Now, we calculate the **probability of enrollment** for lottery winners and losers:

$$P(Enroll_{ij} = 1 | W_{ij} = 1, A_{ij} = 1) = \frac{1+2}{10} = 0.3$$
$$P(Enroll_{ij} = 1 | W_{ij} = 0, A_{ij} = 1) = \frac{1}{10} = 0.1$$

Using the formula for treatment effect (δ):

$$\text{Treatment Effect} = \frac{0.2}{0.3 - 0.1} = 1$$

This confirms that the **true treatment effect** is **1 unit**.

To account for additional factors (X_i), we extend the model as follows:

$$Y_{ia} = \delta W_{ia} + \lambda L_{ia} + X_i\theta + u_{ia}$$

where:

- δ = Intent-to-Treat effect
- λ = **True treatment effect**
- W = Whether a student **wins** the lottery
- L = Whether a student **enrolls** in the school
- $X_i\theta$ = Control variables (i.e., reweighting of lottery), but would not affect treatment effect $E(\delta)$

Since **choosing to apply to a lottery is not random**, we must consider the following:

$$E(\lambda) \neq E(\lambda_1)$$

This demonstrates why **lottery-based assignment** is a useful but imperfect tool for causal inference—winning the lottery is random, but **who applies** is not.

22.5.2 Re-Randomization

In standard randomization, baseline covariates are only balanced on average, meaning imbalance can still occur due to random chance. Re-randomization eliminates bad randomizations by checking balance before the experiment begins (Morgan and Rubin, 2012).

Key Motivations for Re-Randomization

- **Randomization does not guarantee balance**:
  - Example: For **10 covariates**, the probability of at least one imbalance at $\alpha = 0.05$ is:

$$1 - (1 - 0.05)^{10} = 0.40 = 40\%$$

  - This means a high chance of some imbalance across treatment groups.

- **Re-randomization increases precision**: If covariates correlate with the outcome, improving covariate balance improves treatment effect estimation
- **Accounting for re-randomization in inference**: Since re-randomization filters out bad assignments, it is equivalent to increasing the sample size and must be considered when computing standard errors.
- **Alternative balancing techniques**:
  - Stratified randomization (Johansson and Schultzberg, 2022).
  - Matched randomization (Greevy et al., 2004; Kapelner and Krieger, 2014).
  - Minimization (Pocock and Simon, 1975).

Example: Balancing Experimental Groups

An online retailer is testing two website designs (A and B) but wants to ensure that key customer demographics (e.g., age) are balanced across treatment groups.

We define a balance criterion to check if the mean age difference between groups is acceptable before proceeding.

```
set.seed(123)

# Define balance criterion: Ensure mean age difference < 1 year
balance_criterion <- function(data) {
  abs(mean(data$age[data$group == "A"]) - mean(data$age[data$group == "B"])) < 1
}

# Generate randomized groups, repeat until balance criterion is met
```

Figure 22.1: Figure from USC Schaeffer Center

```r
repeat {
  data <- data.frame(
    age = rnorm(100, mean = 35, sd = 10),
    group = sample(c("A", "B"), 100, replace = TRUE)
  )
  if (balance_criterion(data)) break
}

# Check final group means
tapply(data$age, data$group, mean)
#>        A        B
#> 35.91079 35.25483
```

22.5.2.1 Rerandomization Criterion

Re-randomization is based on a function of the covariate matrix ($\mathbf{X}$) and treatment assignments ($\mathbf{W}$).

$$W_i = \begin{cases} 1, & \text{if treated} \\ 0, & \text{if control} \end{cases}$$

A common approach is to use Mahalanobis Distance to measure covariate balance between treatment and control groups:

$$\begin{aligned} M &= (\bar{\mathbf{X}}_T - \bar{\mathbf{X}}_C)' \text{cov}(\bar{\mathbf{X}}_T - \bar{\mathbf{X}}_C)^{-1} (\bar{\mathbf{X}}_T - \bar{\mathbf{X}}_C) \\ &= (\frac{1}{n_T} + \frac{1}{n_C})^{-1} (\bar{\mathbf{X}}_T - \bar{\mathbf{X}}_C)' \text{cov}(\mathbf{X})^{-1} (\bar{\mathbf{X}}_T - \bar{\mathbf{X}}_C) \end{aligned}$$

where:

- $\bar{\mathbf{X}}_T$ and $\bar{\mathbf{X}}_C$ are the mean covariate values for treatment and control groups.
- $\text{cov}(\mathbf{X})$ is the covariance matrix of the covariates.
- n_T and n_C are the sample sizes for treatment and control groups.

If the sample size is large and randomization is pure, then M follows a chi-squared distribution:

$$M \sim \chi^2_k$$

where k is the number of covariates to be balanced.

Choosing the Rerandomization Threshold ($M > a$)

Define p_a as the probability of accepting a randomization:

- Smaller p_a → Stronger balance, but longer computation time.
- Larger p_a → Faster randomization, but weaker balance.

A rule of thumb is to re-randomize whenever:

$$M > a$$

where a is chosen based on acceptable balance thresholds.

We apply Mahalanobis Distance as a balance criterion to ensure that treatment and control groups are well-matched before proceeding with the experiment.

```
set.seed(123)
library(MASS)

# Generate a dataset with two covariates
n <- 100
X <- mvrnorm(n, mu = c(0, 0), Sigma = matrix(c(1, 0.5, 0.5, 1), 2, 2))
colnames(X) <- c("Covariate1", "Covariate2")

# Balance function using Mahalanobis Distance
balance_criterion <- function(X, group) {
  X_treat <- X[group == 1, ]
  X_control <- X[group == 0, ]

  mean_diff <- colMeans(X_treat) - colMeans(X_control)
  cov_inv <- solve(cov(X))
  M <- t(mean_diff) %*% cov_inv %*% mean_diff

  return(M < 3.84)  # Acceptable threshold (chi-squared critical value for k = 2, alpha = 0.05)
}

# Repeat randomization until balance is met
repeat {
  group <- sample(c(0, 1), n, replace = TRUE)
  if (balance_criterion(X, group)) break
}

# Display final balance check
table(group)
#> group
#>  0  1
```

```
#> 50 50
colMeans(X[group == 1, ])  # Treatment group means
#> Covariate1 Covariate2
#>  0.2469635  0.1918521
colMeans(X[group == 0, ])  # Control group means
#>  Covariate1  Covariate2
#>  0.01717088 -0.14281124
```

22.5.3 Two-Stage Randomized Experiments

A **two-stage randomized experiment** involves **sequential interventions**, where treatment assignments depend on earlier responses. This design is widely used in:

- **Adaptive Learning**: Adjusting educational content based on student progress.
- **Personalized Advertising**: Targeting follow-up ads based on engagement.
- **Medical Trials**: Adapting treatments based on patient response.

By introducing **a second randomization stage**, researchers can evaluate:

1. **The effect of initial treatments.**
2. **The effect of follow-up treatments.**
3. **Potential interactions between the two stages.**

22.5.3.1 Example: Personalized Advertising Experiment

A company tests two initial ad campaigns (Ad A vs. Ad B). After observing customer engagement, they apply a second-stage intervention (e.g., Discount vs. No Discount).

The two-stage experiment can be modeled as:

$$Y_{ijk} = \mu + \alpha_i + \beta_{j(i)} + \epsilon_{ijk}$$

where:

- μ = Overall mean outcome (e.g., conversion rate).
- α_i = Effect of first-stage intervention (i = Ad A or Ad B).
- $\beta_{j(i)}$ = Effect of second-stage intervention, nested within first-stage groups.
- ϵ_{ijk} = Random error term.

The nested structure ensures that the second-stage treatment ($\beta_{j(i)}$) is assigned within each first-stage treatment group.

```r
set.seed(123)

# Generate first-stage randomization (Initial Ad)
data <- data.frame(
  stage1 = sample(c("Ad A", "Ad B"), 100, replace = TRUE),
  stage2 = rep(NA, 100)  # Placeholder for second-stage randomization
)

# Second-stage assignment based on first-stage response
data$stage2[data$stage1 == "Ad A"] <-
    sample(c("Discount", "No Discount"),
           sum(data$stage1 == "Ad A"),
           replace = TRUE)
data$stage2[data$stage1 == "Ad B"] <-
    sample(c("Discount", "No Discount"),
           sum(data$stage1 == "Ad B"),
           replace = TRUE)

# Display final randomization
table(data$stage1, data$stage2)
#>
#>        Discount No Discount
#>   Ad A       24          33
#>   Ad B       22          21
```

This structure ensures:

1. **Fair assignment** of initial ads.
2. **Adaptive targeting** in the second stage based on user engagement.

22.5.4 Two-Stage Randomized Experiments with Interference and Noncompliance

In real-world experiments, **interference and noncompliance** complicate analysis (Imai et al., 2021a):

- **Interference**: When treatment effects "spill over" from one group to another (e.g., social influence in marketing).
- **Noncompliance**: When participants do not adhere to their assigned treatment (e.g., a customer ignoring an ad).

To handle **noncompliance**, we define:

- Z_{ik} = **Assigned treatment** (e.g., Ad A or Ad B).
- D_{ik} = **Actual treatment received** (e.g., whether the user actually saw the ad).

- Y_{ik} = **Outcome** (e.g., purchase).

A **two-stage Instrumental Variable model** adjusts for noncompliance:

$$D_{ik} = \gamma_0 + \gamma_1 Z_{ik} + v_{ik}$$
$$Y_{ik} = \beta_0 + \beta_1 D_{ik} + \epsilon_{ik}$$

where:

- γ_1 measures the effect of assignment on actual treatment received.
- β_1 estimates the treatment effect, adjusting for noncompliance.

If individuals influence each other's outcomes, traditional randomization is biased. Solutions include:

1. **Cluster Randomization**: Assigning treatments at the group level (e.g., entire social circles receive the same ad).
2. **Partial Interference Models**: Assume interference only occurs within predefined groups.

```r
set.seed(123)
library(ivreg)  # Load Instrumental Variable Regression Package

# Generate data for first-stage treatment assignment
n <- 500
data <- data.frame(
  Z = sample(c(0, 1), n, replace = TRUE),  # Initial assignment (randomized)
  D = NA,  # Actual treatment received (affected by compliance)
  Y = NA   # Outcome variable (e.g., purchase)
)

# Introduce noncompliance: 80% compliance rate
data$D <- ifelse(runif(n) < 0.8, data$Z, 1 - data$Z)

# Generate outcome variable (Y) with true treatment effect
data$Y <- 5 + 3 * data$D + rnorm(n, mean = 0, sd = 2)  # True effect of D on Y is 3

# Estimate Two-Stage Least Squares (2SLS)
iv_model <- ivreg(Y ~ D | Z, data = data)
summary(iv_model)
#>
#> Call:
#> ivreg(formula = Y ~ D | Z, data = data)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -5.497 -1.344  0.018  1.303  5.493
```

```
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   5.1285     0.1861  27.557   <2e-16 ***
#> D             2.7487     0.3072   8.949   <2e-16 ***
#>
#> Diagnostic tests:
#>                  df1 df2 statistic p-value
#> Weak instruments   1 498    263.54  <2e-16 ***
#> Wu-Hausman         1 497      0.19   0.663
#> Sargan             0  NA        NA      NA
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 2.017 on 498 degrees of freedom
#> Multiple R-Squared: 0.2997,  Adjusted R-squared: 0.2983
#> Wald test: 80.08 on 1 and 498 DF,  p-value: < 2.2e-16
```

- The first-stage regression estimates how strongly Z affects D (compliance).
- The second-stage regression estimates the true causal effect of D on Y.

If interference is present, the standard IV method may be biased. Researchers should explore network-based randomization or spatial models.

22.6 Emerging Research

Recent research highlights significant challenges in measuring the causal effects of ad content in online advertising experiments. Digital advertising platforms employ algorithmic targeting and dynamic ad delivery mechanisms, which can introduce systematic biases in A/B testing (Braun and Schwartz, 2025). Key concerns include:

- **Nonrandom Exposure**: Online advertising platforms optimize ad delivery dynamically, meaning users are not randomly assigned to different ad variants. Instead, platforms use proprietary algorithms to serve ads to different, often highly heterogeneous, user groups.
- **Divergent Delivery**: The optimization process can lead to "divergent delivery," where differences in user engagement patterns and platform objectives result in non-comparable treatment groups. This confounds the true effect of ad content with algorithmic biases in exposure.
- **Bias in Measured Effects**: Algorithmic targeting, user heterogeneity, and data aggregation can distort both the **magnitude** and **direction** of estimated ad effects. This means traditional A/B test results may not accurately reflect the true impact of ad creatives.

- **Limited Transparency**: Platforms have little incentive to assist advertisers or researchers in disentangling ad content effects from proprietary targeting mechanisms. As a result, experimenters must design careful identification strategies to mitigate these biases.

22.6.1 Handling Zero-Valued Outcomes

When analyzing treatment effects, a common issue arises when the outcome variable includes **zero values**.

For example, in business applications, a marketing intervention may have **no effect on some customers** (resulting in zero sales). If we were to apply a log transformation to the outcome variable, this would be problematic because:

- log(0) is undefined.
- Log transformation is **sensitive to outcome units**, making interpretation difficult (Chen and Roth, 2023).

Instead of applying a log transformation, we should use methods that are **robust to zero values**:

1. **Percentage changes in the Average**
   - By using Poisson Quasi-Maximum Likelihood Estimation (QMLE), we can interpret coefficients as percentage changes in the mean outcome of the treatment group relative to the control group.
2. **Extensive vs. Intensive Margins**
   - This approach distinguishes between:
     - **Extensive margin**: The likelihood of moving from zero to a positive outcome (e.g., increasing the probability of making a sale).
     - **Intensive margin**: The increase in the outcome given that it is already positive (e.g., increasing the sales amount).
   - To estimate the **intensive-margin bounds**, we could use Lee (2009), assuming that treatment has a **monotonic effect** on the outcome.

We first generate a dataset to simulate a scenario where an outcome variable (e.g., sales, website clicks) has **many zeros**, and treatment affects both the **extensive** and **intensive margins**.

```r
set.seed(123) # For reproducibility
library(tidyverse)

n <- 1000 # Number of observations
p_treatment <- 0.5 # Probability of being treated

# Step 1: Generate the treatment variable D
D <- rbinom(n, 1, p_treatment)
```

```r
# Step 2: Generate potential outcomes
# Untreated potential outcome (mostly zeroes)
Y0 <- rnorm(n, mean = 0, sd = 1) * (runif(n) < 0.3)

# Treated potential outcome (affecting both extensive and intensive margins)
Y1 <- Y0 + rnorm(n, mean = 2, sd = 1) * (runif(n) < 0.7)

# Step 3: Combine effects based on treatment assignment
Y_observed <- (1 - D) * Y0 + D * Y1

# Ensure non-negative outcomes (modeling real-world situations)
Y_observed[Y_observed < 0] <- 0

# Create a dataset with an additional control variable
data <- data.frame(
    ID = 1:n,
    Treatment = D,
    Outcome = Y_observed,
    X = rnorm(n) # Control variable
) |>
    dplyr::mutate(positive = Outcome > 0) # Indicator for extensive margin

# View first few rows
head(data)
#>   ID Treatment   Outcome          X positive
#> 1  1         0 0.0000000  1.4783345    FALSE
#> 2  2         1 2.2369379 -1.4067867     TRUE
#> 3  3         0 0.0000000 -1.8839721    FALSE
#> 4  4         1 3.2192276 -0.2773662     TRUE
#> 5  5         1 0.6649693  0.4304278     TRUE
#> 6  6         0 0.0000000 -0.1287867    FALSE

# Plot distribution of outcomes
hist(
    data$Outcome,
    breaks = 30,
    main = "Distribution of Outcomes",
    col = "lightblue"
)
```

22.6.1.1 Estimating Percentage Changes in the Average

Since we cannot use a log transformation, we estimate **percentage changes** using **Poisson QMLE**, which is robust to zero-valued outcomes.

```
library(fixest)

# Poisson Quasi-Maximum Likelihood Estimation (QMLE)
res_pois <- fepois(
    fml = Outcome ~ Treatment + X,
    data = data,
    vcov = "hetero"
)

# Display results
etable(res_pois)
#>                            res_pois
#> Dependent Var.:             Outcome
#>
#> Constant        -2.223*** (0.1440)
#> Treatment        2.579*** (0.1494)
#> X                  0.0235 (0.0406)
#> _______________ __________________
#> S.E. type       Heteroskedas.-rob.
#> Observations                 1,000
#> Squared Cor.               0.33857
#> Pseudo R2                  0.26145
```

```
#> BIC                        1,927.9
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To interpret the results:

- The coefficient on `Treatment` represents the **log-percentage change** in the mean outcome of the treatment group relative to the control group.
- To compute the **proportional effect**, we exponentiate the coefficient:

```
# Compute the proportional effect of treatment
treatment_effect <- exp(coefficients(res_pois)["Treatment"]) - 1
treatment_effect
#> Treatment
#>  12.17757
```

Compute the **standard error**:

```
# Compute standard error
se_treatment <- exp(coefficients(res_pois)["Treatment"]) *
    sqrt(res_pois$cov.scaled["Treatment", "Treatment"])
se_treatment
#> Treatment
#>  1.968684
```

Thus, we conclude that the treatment effect increases the outcome by **1215%** for the treated group compared to the control group.

22.6.1.2 Estimating Extensive vs. Intensive Margins

We now estimate **treatment effects separately** for the **extensive margin** (probability of a nonzero outcome) and the **intensive margin** (magnitude of effect among those with a nonzero outcome).

First, we use **Lee Bounds** to estimate the **intensive-margin effect** for individuals who always have a nonzero outcome.

```
library(causalverse)

res <- causalverse::lee_bounds(
    df = data,
    d = "Treatment",
    m = "positive",
    y = "Outcome",
    numdraws = 10
) |>
    causalverse::nice_tab(2)
```

```r
print(res)
#>          term estimate std.error
#> 1 Lower bound    -0.22      0.09
#> 2 Upper bound     2.77      0.14
```

Since the confidence interval **includes zero**, we cannot conclude that the treatment has a significant **intensive-margin effect**.

22.6.1.3 Sensitivity Analysis: Varying the Effect for Compliers

To further investigate the intensive-margin effect, we consider **how sensitive our results are** to different assumptions about compliers.

We assume that the expected outcome for compliers is $100 \times c$ **lower or higher** than for always-takers:

$$E(Y(1)\|\text{Complier}) = (1 - c)E(Y(1)\|\text{Always-taker})$$

We compute Lee Bounds for different values of c:

```r
set.seed(1)
c_values = c(0.1, 0.5, 0.7)

combined_res <- bind_rows(lapply(c_values, function(c) {
    res <- causalverse::lee_bounds(
        df = data,
        d = "Treatment",
        m = "positive",
        y = "Outcome",
        numdraws = 10,
        c_at_ratio = c
    )

    res$c_value <- as.character(c)
    return(res)
}))

combined_res |>
    dplyr::select(c_value, everything()) |>
    causalverse::nice_tab()
#>   c_value           term estimate std.error
#> 1     0.1 Point estimate     6.60      0.71
#> 2     0.5 Point estimate     2.54      0.13
#> 3     0.7 Point estimate     1.82      0.08
```

- If we assume $c = 0.1$, meaning compliers' outcomes are 10% of always-takers', then the intensive-margin effect is 6.6 units higher for always-takers.

- If $c = 0.5$, meaning compliers' outcomes are 50% of always-takers', then the intensive-margin effect is 2.54 units higher.

These results highlight how **assumptions about compliers affect conclusions** about the intensive margin.

When dealing with **zero-valued outcomes**, log transformations are **not appropriate**. Instead:

- **Poisson QMLE** provides a robust way to estimate **percentage changes** in the outcome.
- **Extensive vs. Intensive Margins** allow us to distinguish between:
  - The **probability** of a nonzero outcome (**extensive margin**).
  - The **magnitude** of change among those with a nonzero outcome (**intensive margin**).
- **Lee Bounds** provide a method to estimate the **intensive-margin effect**, though results can be **sensitive to assumptions** about always-takers and compliers.

Chapter 23

Sampling

Sampling allows us to draw conclusions about a population without analyzing every individual in it. In business applications—such as marketing research, and financial forecasting—sampling enables efficient decision-making while reducing costs and effort.

23.1 Population and Sample

This is a refresher on terminology regarding sampling.

- **Population** (N): The complete set of all elements under study.
- **Sample** (n): A subset of the population selected for analysis.
- **Parameter**: A numerical measure that describes a characteristic of a population (e.g., population mean μ, population variance σ^2).
- **Statistic**: A numerical measure computed from a sample, used to estimate a population parameter (e.g., sample mean $\bar{x}$, sample variance s^2).

A well-chosen sample ensures that results generalize to the population, reducing **sampling bias**.

23.2 Sampling Techniques

23.2.1 Probability Sampling

Probability sampling methods ensure that every element in the population has a known, nonzero probability of being selected. These methods are preferred in inferential statistics since they allow for the estimation of sampling error.

23.2.2 Simple Random Sampling

Simple Random Sampling (SRS) ensures that every element in the population has an **equal chance** of being selected. This can be done **with replacement** or **without replacement**, impacting whether an element can be chosen more than once.

Below is an example of drawing a simple random sample without replacement from a population of 100 elements:

```
set.seed(123)
population <- 1:100  # A population of 100 elements
sample_srs <- sample(population, size = 10, replace = FALSE)
sample_srs
#>  [1] 31 79 51 14 67 42 50 43 97 25
```

Advantages:

- Simple and easy to implement
- Ensures unbiased selection

Disadvantages:

- May not represent subgroups well, especially in heterogeneous populations
- Requires access to a complete list of the population

23.2.2.1 Using `dplyr`

The `sample_n()` function in `dplyr` allows for simple random sampling from a dataset:

```
library(dplyr)
iris_df <- iris
set.seed(1)
sample_n(iris_df, 5)  # Randomly selects 5 rows from the iris dataset
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
#> 1          5.8         2.7          4.1         1.0 versicolor
#> 2          6.4         2.8          5.6         2.1  virginica
#> 3          4.4         3.2          1.3         0.2     setosa
#> 4          4.3         3.0          1.1         0.1     setosa
#> 5          7.0         3.2          4.7         1.4 versicolor
```

23.2.2.2 Using the `sampling` Package

The `sampling` package provides functions for random sampling **with** and **without** replacement.

```
library(sampling)
# Assign a unique ID to each row in the dataset
```

```
iris_df$id <- 1:nrow(iris_df)

# Simple random sampling without replacement
srs_sample <- srswor(10, length(iris_df$id))
# srs_sample

# Simple random sampling with replacement
srs_sample_wr <- srswr(10, length(iris_df$id))
# srs_sample_wr
```

23.2.2.3 Using the sampler Package

The `sampler` package provides additional functionality, such as **oversampling** to account for non-response.

```
library(sampler)
rsamp(albania, n = 260, over = 0.1, rep = FALSE)
```

23.2.2.4 Handling Missing Data in Sample Collection

To compare a sample with received (collected) data and identify missing elements:

```
alsample <- rsamp(df = albania, 544)  # Initial sample
alreceived <- rsamp(df = alsample, 390)  # Collected data
rmissing(sampdf = alsample, colldf = alreceived, col_name = qvKod)
```

23.2.3 Stratified Sampling

Stratified sampling involves dividing the population into distinct **strata** based on a characteristic (e.g., age, income level, region). A **random sample** is then drawn from each stratum, often in proportion to its size within the population. This method ensures that all subgroups are adequately represented, improving the precision of estimates.

The following example demonstrates stratified sampling where individuals belong to three different groups **(A, B, C)**, and a random sample is drawn from each.

```
library(dplyr)

set.seed(123)
data <- data.frame(
  ID = 1:100,
  Group = sample(c("A", "B", "C"), 100, replace = TRUE)
```

```
)

# Stratified random sampling: selecting 10 elements per group
stratified_sample <- data %>%
  group_by(Group) %>%
  sample_n(size = 10)

# stratified_sample
```

Advantages:

- Ensures representation of all subgroups
- More precise estimates compared to Simple Random Sampling
- Reduces sampling error by accounting for population variability

Disadvantages:

- Requires prior knowledge of population strata
- More complex to implement than SRS

23.2.3.1 Using `dplyr` for Stratified Sampling

Sampling by Fixed Number of Rows

Here, we extract **5 random observations** from each species in the `iris` dataset.

```
library(dplyr)

set.seed(123)
sample_iris <- iris %>%
  group_by(Species) %>%
  sample_n(5)  # Selects 5 samples per species

# sample_iris
```

Sampling by Fraction of Each Stratum

Instead of selecting a fixed number, we can sample **15% of each species**:

```
set.seed(123)
sample_iris <- iris %>%
  group_by(Species) %>%
  sample_frac(size = 0.15)  # Selects 15% of each species

# sample_iris
```

23.2.3.2 Using the sampler Package

The sampler package allows stratified sampling with proportional allocation:

```
library(sampler)

# Stratified sample using proportional allocation without replacement
ssamp(df = albania, n = 360, strata = qarku, over = 0.1)
```

23.2.3.3 Handling Missing Data in Stratified Sampling

To identify the number of missing values **by stratum** between the initial sample and the collected data:

```
alsample <- rsamp(df = albania, 544)  # Initial sample
alreceived <- rsamp(df = alsample, 390)  # Collected data

smissing(
  sampdf = alsample,
  colldf = alreceived,
  strata = qarku,   # Strata column
  col_name = qvKod  # Column for checking missing values
)
```

23.2.4 Systematic Sampling

Selects every kth element after a random starting point.

```
k <- 10  # Select every 10th element
start <- sample(1:k, 1)  # Random start point
sample_systematic <- population[seq(start, length(population), by = k)]
```

Advantages:

- Simple to implement
- Ensures even coverage

Disadvantages:

- If data follows a pattern, bias may be introduced

23.2.5 Cluster Sampling

Instead of selecting individuals, entire clusters (e.g., cities, schools) are randomly chosen, and all members of selected clusters are included.

```
data$Cluster <- sample(1:10, 100, replace = TRUE)  # Assign 10 clusters
chosen_clusters <- sample(1:10, size = 3)  # Select 3 clusters
cluster_sample <- filter(data, Cluster %in% chosen_clusters)
```

Advantages:

- Cost-effective when the population is large
- Useful when the population is naturally divided into groups

Disadvantages:

- Higher variability
- Risk of unrepresentative clusters

23.2.6 Non-Probability Sampling

These methods do not give all elements a known probability of selection. They are used in exploratory research but are not suitable for making formal statistical inferences.

23.2.7 Convenience Sampling

Selecting individuals who are easiest to reach (e.g., mall surveys).

Pros: Quick and inexpensive
Cons: High risk of bias, not generalizable

23.2.8 Quota Sampling

Similar to stratified sampling but non-random.

Pros: Ensures subgroup representation
Cons: Subject to selection bias

23.2.9 Snowball Sampling

Used for hard-to-reach populations (e.g., networking through referrals).

Pros: Useful when the population is unknown
Cons: High bias, dependency on initial subjects

23.3 Unequal Probability Sampling

Unequal probability sampling assigns different selection probabilities to elements in the population. This approach is often used when certain units are more important, have higher variability, or require higher precision in estimation.

Common methods for unequal probability sampling include:

- **Probability Proportional to Size (PPS)**: Selection probability is proportional to a given auxiliary variable (e.g., revenue, population size).

- **Poisson Sampling**: Independent selection of each unit with a given probability.
- **Systematic Sampling with Unequal Probabilities**: Uses a systematic approach while ensuring different probabilities.

The following functions from the `sampling` package implement various unequal probability sampling methods:

```
library(sampling)

# Different methods for unequal probability sampling
UPbrewer()          # Brewer's method
UPmaxentropy()      # Maximum entropy method
UPmidzuno()         # Midzuno's method
UPmidzunopi2()      # Midzuno's method with second-order inclusion probabilities
UPmultinomial()     # Multinomial method
UPpivotal()         # Pivotal method
UPrandompivotal()   # Randomized pivotal method
UPpoisson()         # Poisson sampling
UPsampford()        # Sampford's method
UPsystematic()      # Systematic sampling
UPrandomsystematic() # Randomized systematic sampling
UPsystematicpi2()   # Systematic sampling with second-order probabilities
UPtille()           # Tillé's method
UPtillepi2()        # Tillé's method with second-order inclusion probabilities
```

Each of these methods has specific use cases and theoretical justifications. For example:

- **Poisson sampling** allows flexible control over sample size but may lead to variable sample sizes.
- **Systematic sampling** is useful when population elements are arranged in a meaningful order.
- **Tillé's method** ensures better control over the sample's second-order inclusion probabilities.

23.4 Balanced Sampling

Balanced sampling ensures that the **means of auxiliary variables** in the sample match those in the population. This method improves estimation efficiency and reduces variability without introducing bias.

Balanced sampling **differs from purposive selection** because it still involves **randomization**, ensuring statistical validity.

The **balancing equation** is given by:

$$\sum_{k \in S} \frac{\mathbf{x}_k}{\pi_k} = \sum_{k \in U} \mathbf{x}_k$$

where:

- $\mathbf{x}_k$ is a vector of auxiliary variables (e.g., income, age, household size).
- π_k is the inclusion probability of unit k.
- S is the sample, and U is the population.

This ensures that the total weighted sum of auxiliary variables in the sample matches the total sum in the population.

23.4.1 Cube Method for Balanced Sampling

The **Cube Method** is a widely used approach for balanced sampling, consisting of two phases:

1. **Flight Phase**: Ensures initial balance on auxiliary variables.
2. **Landing Phase**: Adjusts the sample to meet constraints while keeping randomness.

```r
library(sampling)

# Cube method functions
samplecube()        # Standard cube method
fastflightcube()    # Optimized flight phase
landingcube()       # Landing phase method
```

23.4.2 Balanced Sampling with Stratification

Stratification attempts to **replicate the population structure** in the sample by preserving the original **multivariate histogram**.

```r
library(survey)
data("api")

# Stratified design with proportional allocation
srs_design <- svydesign(data = apistrat,
                        weights = ~pw,
                        fpc = ~fpc,
                        strata = ~stype,
                        id = ~1)
```

An additional method for balanced stratification is:

```
balancedstratification()
```

This method ensures that within each stratum, the sample retains the original proportions of auxiliary variables.

23.4.3 Balanced Sampling in Cluster Sampling

Cluster sampling involves selecting **entire groups (clusters)** instead of individual units. A balanced approach ensures that the sampled clusters **preserve the overall distribution of auxiliary variables**.

```
library(survey)
data("api")

# Cluster sampling design
srs_design <- svydesign(data = apiclus1,
                        weights = ~pw,
                        fpc = ~fpc,
                        id = ~dnum)
```

For explicitly balanced cluster sampling:

```
balancedcluster()
```

This method ensures that the **cluster-level characteristics** of the sample match those of the population.

23.4.4 Balanced Sampling in Two-Stage Sampling

Two-stage sampling first selects primary units (e.g., schools, cities) and then samples within them. A balanced approach ensures **representative selection at both stages**.

```
library(survey)
data("api")

# Two-stage sampling design
srs_design <- svydesign(data = apiclus2,
                        fpc = ~fpc1 + fpc2,
                        id = ~dnum + snum)
```

For explicitly balanced two-stage sampling:

```
balancedtwostage()
```

This method ensures that **auxiliary variables remain balanced across both selection stages**, reducing variability while maintaining randomness.

23.5 Sample Size Determination

The appropriate sample size depends on the **margin of error**, **confidence level**, and **population variability**. A commonly used formula for estimating the required sample size for a proportion is:

$$n = \frac{Z^2 p(1-p)}{E^2}$$

where:

- Z is the Z-score corresponding to the confidence level
- p is the estimated proportion
- E is the margin of error

```
z <- qnorm(0.975)  # 95% confidence level
p <- 0.5  # Estimated proportion
E <- 0.05  # 5% margin of error
n <- (z^2 * p * (1 - p)) / (E^2)
ceiling(n)  # Round up to nearest integer
#> [1] 385
```

Chapter 24

Analysis of Variance

Analysis of Variance (ANOVA) shares its underlying mechanism with linear regression. However, ANOVA approaches the analysis from a different perspective, making it particularly useful for studying **qualitative variables** and **designed experiments**.

Key Terminology

- **Factor**: An explanatory or predictor variable studied in an experiment.
- **Treatment (Factor Level)**: A specific value or category of a factor applied to an experimental unit.
- **Experimental Unit**: The entity (e.g., person, animal, material) subjected to treatments and providing a response.
- **Single-Factor Experiment**: An experiment with only one explanatory variable.
- **Multifactor Experiment**: An experiment involving multiple explanatory variables.
- **Classification Factor**: A factor that is not controlled by the experimenter (common in observational studies).
- **Experimental Factor**: A factor that is directly assigned by the experimenter.

A well-designed experiment requires careful planning in the following areas:

- **Choice of treatments**: Selecting factor levels to be tested.
- **Selection of experimental units**: Ensuring an appropriate sample.
- **Treatment assignment**: Avoiding selection bias through proper randomization.
- **Measurement**: Minimizing measurement bias and considering blinding when necessary.

Advancements in Experimental Design

1. **Factorial Experiments**:
   - Investigate multiple factors simultaneously.
   - Allow for the study of **interactions** between factors.
2. **Replication**:
   - Repeating experiments increases statistical power.
   - Helps estimate **mean squared error**.
3. **Randomization**:
   - Introduced formally by R.A. Fisher in the early 1900s.
   - Ensures that treatment assignment is not systematically biased.
   - Helps eliminate confounding effects due to time, space, or other lurking variables.
4. **Local Control (Blocking/Stratification)**:
   - Reduces experimental error by controlling for known sources of variability.
   - Increases power by grouping similar experimental units together before randomizing treatments.

Randomization also helps eliminate correlations due to time and space.

24.1 Completely Randomized Design

A **Completely Randomized Design (CRD)** is the simplest type of experimental design, where experimental units are randomly assigned to treatments.

Consider a treatment factor A with $a \geq 2$ treatment levels. Each experimental unit is randomly assigned to one of these levels. The number of units in each group can be:

- **Balanced**: All groups have equal sample sizes n.
- **Unbalanced**: Groups have different sample sizes n_i (for $i = 1, ..., a$).

The total sample size is given by:

$$N = \sum_{i=1}^{a} n_i$$

The number of possible assignments of units to treatments is:

$$k = \frac{N!}{n_1! n_2! \dots n_a!}$$

Each assignment has an equal probability of being selected: $1/k$. The response of each experimental unit is denoted as Y_{ij}, where:

- i indexes the treatment group.

- j indexes the individual unit within treatment i.

Table 24.1: Treatment Response Table

Treatment	1	2	...	a
	Y_{11}	Y_{21}	...	Y_{a1}
	Y_{12}	...	...	...
	...	...	...	...
Sample Mean	$\bar{Y}_{1.}$	$\bar{Y}_{2.}$	...	$\bar{Y}_{a.}$
Sample SD	s_1	s_2	...	s_a

Where:

$$\bar{Y}_{i.} = \frac{1}{n_i} \sum_{j=1}^{n_i} Y_{ij}$$

$$s_i^2 = \frac{1}{n_i - 1} \sum_{j=1}^{n_i} (Y_{ij} - \bar{Y}_{i.})^2$$

The **grand mean** is:

$$\bar{Y}_{..} = \frac{1}{N} \sum_i \sum_j Y_{ij}$$

24.2 Single-Factor Fixed Effects Model

Also known as **One-Way ANOVA** or **ANOVA Type I Model**.

24.2.1 Partitioning the Variance

The total variability in the response variable Y_{ij} can be decomposed as follows:

$$\begin{aligned} Y_{ij} - \bar{Y}_{..} &= Y_{ij} - \bar{Y}_{..} + \bar{Y}_{i.} - \bar{Y}_{i.} \\ &= (\bar{Y}_{i.} - \bar{Y}_{..}) + (Y_{ij} - \bar{Y}_{i.}) \end{aligned}$$

where:

- The first term represents **between-treatment variability** (deviation of treatment means from the grand mean).
- The second term represents **within-treatment variability** (deviation of observations from their treatment mean).

Thus, we partition the **total sum of squares (SSTO)** as:

$$\sum_i \sum_j (Y_{ij} - \bar{Y}_{..})^2 = \sum_i n_i(\bar{Y}_{i.} - \bar{Y}_{..})^2 + \sum_i \sum_j (Y_{ij} - \bar{Y}_{i.})^2$$

Or equivalently:

$$SSTO = SSTR + SSE$$

Where:

- **SSTO (Total SS)**: Total variability in the data.
- **SSTR (Treatment SS)**: Variability due to differences between treatment means.
- **SSE (Error SS)**: Variability within treatments (unexplained variance).

Degrees of freedom (d.f.):

$$(N-1) = (a-1) + (N-a)$$

where we lose a degree of freedom for the total corrected SSTO because of the estimation of the mean ($\sum_i \sum_j (Y_{ij} - \bar{Y}_{..}) = 0$) and for the SSTR ($\sum_i n_i(\bar{Y}_{i.} - \bar{Y}_{..}) = 0$)

Mean squares:

$$MSTR = \frac{SSTR}{a-1}, \quad MSR = \frac{SSE}{N-a}$$

Table 24.2: ANOVA Table

Source of Variation	SS	df	MS
Between Treatments	$\sum_i n_i(\bar{Y}_{i.} - \bar{Y}_{..})^2$	$a-1$	$SSTR/(a-1)$
Error (within treatments)	$\sum_i \sum_j (Y_{ij} - \bar{Y}_{i.})^2$	$N-a$	$SSE/(N-a)$
Total (corrected)	$\sum_i n_i(\bar{Y}_{i.} - \bar{Y}_{..})^2$	$N-1$	

24.3 Linear Model Interpretation of ANOVA

24.3.1 Cell Means Model

The **cell means model** describes the response as:

$$Y_{ij} = \mu_i + \epsilon_{ij}$$

where:

- Y_{ij}: Response for unit j in treatment i.
- μ_i: Fixed population mean for treatment i.
- $\epsilon_{ij} \sim N(0, \sigma^2)$: Independent errors.
- $E(Y_{ij}) = \mu_i$, $\text{Var}(Y_{ij}) = \sigma^2$.

All observations are assumed to have **equal variance** across treatments.

Example: ANOVA with $a = 3$ Treatments

Consider a case with **three treatments** ($a = 3$), where each treatment has **two replicates** ($n_1 = n_2 = n_3 = 2$). The response vector can be expressed in matrix form as:

$$\begin{pmatrix} Y_{11} \\ Y_{12} \\ Y_{21} \\ Y_{22} \\ Y_{31} \\ Y_{32} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{pmatrix} + \begin{pmatrix} \epsilon_{11} \\ \epsilon_{12} \\ \epsilon_{21} \\ \epsilon_{22} \\ \epsilon_{31} \\ \epsilon_{32} \end{pmatrix}$$

$$\mathbf{y} = \mathbf{X} \ +$$

where:

- $X_{k,ij} = 1$ if the k-th treatment is applied to unit (i, j).
- $X_{k,ij} = 0$ otherwise.

Note: There is **no intercept** term in this model.

The least squares estimator for β is given by:

$$
\begin{aligned}
\mathbf{b} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} &= (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \\
&= \begin{bmatrix} n_1 & 0 & 0 \\ 0 & n_2 & 0 \\ 0 & 0 & n_3 \end{bmatrix}^{-1} \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} \qquad (24.1) \\
&= \begin{bmatrix} \bar{Y}_1 \\ \bar{Y}_2 \\ \bar{Y}_3 \end{bmatrix}
\end{aligned}
$$

Thus, the estimated treatment means are:

$$\hat{\mu}_i = \bar{Y}_i, \quad i = 1, 2, 3$$

This estimator $\mathbf{b} = [\bar{Y}_1, \bar{Y}_2, \bar{Y}_3]'$ is the **best linear unbiased estimator (BLUE)** for β (i.e., $E(\mathbf{b}) = \beta$)

Since $\mathbf{b} \sim N(\beta, \sigma^2(\mathbf{X}'\mathbf{X})^{-1})$, the variance of the estimated treatment means is:

$$var(\mathbf{b}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-1} = \sigma^2 \begin{bmatrix} 1/n_1 & 0 & 0 \\ 0 & 1/n_2 & 0 \\ 0 & 0 & 1/n_3 \end{bmatrix}$$

Thus, the variance of each estimated treatment mean is:

$$var(b_i) = var(\hat{\mu}_i) = \frac{\sigma^2}{n_i}, \quad i = 1, 2, 3$$

The **mean squared error (MSE)** is given by:

$$
\begin{aligned}
MSE &= \frac{1}{N-a} \sum_i \sum_j (Y_{ij} - \bar{Y}_{i.})^2 \\
&= \frac{1}{N-a} \sum_i [(n_i - 1)s_i^2] \\
&= \frac{1}{N-a} \sum_i (n_i - 1)s_i^2
\end{aligned}
$$

where s_i^2 is the sample variance within the i-th treatment group.

Since $E(s_i^2) = \sigma^2$, we get:

$$E(MSE) = \frac{1}{N-a} \sum_i (n_i - 1)\sigma^2 = \sigma^2$$

Thus, **MSE is an unbiased estimator of** σ^2, regardless of whether the treatment means are equal.

24.4 Expected Mean Square for Treatments

The expected mean square for treatments (MSTR) is:

$$E(MSTR) = \sigma^2 + \frac{\sum_i n_i(\mu_i - \mu_.)^2}{a-1}$$

where:

$$\mu_. = \frac{\sum_{i=1}^{a} n_i \mu_i}{\sum_{i=1}^{a} n_i}$$

If all treatment means are equal ($\mu_1 = \mu_2 = \cdots = \mu_a = \mu_.$), then:

$$E(MSTR) = \sigma^2$$

24.5 F-Test for Equality of Treatment Means

We test the null hypothesis:

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_a$$

against the alternative:

$$H_a : \text{at least one } \mu_i \text{ differs}$$

The **test statistic** is:

$$F = \frac{MSTR}{MSE}$$

- Large values of F suggest rejecting H_0 (since MSTR will be larger than MSE when H_a is true).
- Values of F near 1 suggest that we fail to reject H_0.

Since $MSTR$ and MSE are independent chi-square random variables scaled by their degrees of freedom, under H_0:

$$F \sim F_{(a-1, N-a)}$$

24.5.1 Decision Rule:

- If $F \leq F_{(a-1,N-a;1-\alpha)}$, **fail to reject** H_0.
- If $F \geq F_{(a-1,N-a;1-\alpha)}$, **reject** H_0.

24.5.2 Special Case: $a = 2$

If there are only two treatments ($a = 2$), the ANOVA F-test **reduces to the two-sample** t-test:

$$F = t^2$$

where:

$$t = \frac{\bar{Y}_1 - \bar{Y}_2}{\sqrt{MSE\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}$$

Example:

$a = 3$ (3 treatments) $n_1 = n_2 = n_3 = 2$

$$\begin{pmatrix} Y_{11} \\ Y_{12} \\ Y_{21} \\ Y_{22} \\ Y_{31} \\ Y_{32} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{pmatrix} + \begin{pmatrix} \epsilon_{11} \\ \epsilon_{12} \\ \epsilon_{21} \\ \epsilon_{22} \\ \epsilon_{31} \\ \epsilon_{32} \end{pmatrix}$$

$$\mathbf{y} = \mathbf{X} \ +$$

$X_{k,ij} = 1$ if the k-th treatment is used

$X_{k,ij} = 0$ Otherwise

Note: no intercept term.

$$\begin{aligned} \mathbf{b} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} &= (\mathbf{x}'\mathbf{x})^{-1}\mathbf{x}'\mathbf{y} \\ &= \begin{bmatrix} n_1 & 0 & 0 \\ 0 & n_2 & 0 \\ 0 & 0 & n_3 \end{bmatrix}^{-1} \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} \\ &= \begin{bmatrix} \bar{Y}_1 \\ \bar{Y}_2 \\ \bar{Y}_3 \end{bmatrix} \end{aligned} \tag{24.2}$$

is the BLUE (best linear unbiased estimator) for $\beta = [\mu_1 \mu_2 \mu_3]'$

$$E(\mathbf{b}) = \beta$$

$$var(\mathbf{b}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-1} = \sigma^2 \begin{bmatrix} 1/n_1 & 0 & 0 \\ 0 & 1/n_2 & 0 \\ 0 & 0 & 1/n_3 \end{bmatrix}$$

$var(b_i) = var(\hat{\mu}_i) = \sigma^2/n_i$ where $\mathbf{b} \sim N(\beta, \sigma^2(\mathbf{X}'\mathbf{X})^{-1})$

$$\begin{aligned} MSE &= \frac{1}{N-a} \sum_i \sum_j (Y_{ij} - \bar{Y}_{i.})^2 \\ &= \frac{1}{N-a} \sum_i [(n_i - 1) \frac{\sum_i (Y_{ij} - \bar{Y}_{i.})^2}{n_i - 1}] \\ &= \frac{1}{N-a} \sum_i (n_i - 1) s_1^2 \end{aligned}$$

We have $E(s_i^2) = \sigma^2$

$E(MSE) = \frac{1}{N-a} \sum_i (n_i - 1)\sigma^2 = \sigma^2$

Hence, MSE is an unbiased estimator of σ^2, regardless of whether the treatment means are equal or not.

$E(MSTR) = \sigma^2 + \frac{\sum_i n_i(\mu_i - \mu_.)^2}{a-1}$
where $\mu_. = \frac{\sum_{i=1}^a n_i \mu_i}{\sum_{i=1}^a n_i}$
If all treatment means are equals ($=\mu_.$), $E(MSTR) = \sigma^2$.

Then we can use an F-test for the equality of all treatment means:

$$H_0 : \mu_1 = \mu_2 = .. = \mu_a$$

$$H_a : \textit{not all } \mu_i \textit{ are equal}$$

$F = \frac{MSTR}{MSE}$
where large values of F support H_a (since MSTR will tend to exceed MSE when H_a holds)
and F near 1 support H_0 (upper tail test)

Equivalently, when H_0 is true, $F \sim f_{(a-1, N-a)}$

- If $F \leq f_{(a-1, N-a; 1-\alpha)}$, we cannot reject H_0
- If $F \geq f_{(a-1, N-a; 1-\alpha)}$, we reject H_0

Note: If $a = 2$ (2 treatments), F-test = two sample t-test

24.5.2.1 Treatment Effects (Factor Effects)

Besides Cell means model, we have another way to formalize one-way ANOVA:

$$Y_{ij} = \mu + \tau_i + \epsilon_{ij}$$

where

- Y_{ij} is the j-th response for the i-th treatment
- τ_i is i-th treatment effect
- μ constant component, common to all observations
- ϵ_{ij} independent random errors $\sim N(0, \sigma^2)$

For example, $a = 3$, $n_1 = n_2 = n_3 = 2$

$$\begin{pmatrix} Y_{11} \\ Y_{12} \\ Y_{21} \\ Y_{22} \\ Y_{31} \\ Y_{32} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mu \\ \tau_1 \\ \tau_2 \\ \tau_3 \end{pmatrix} + \begin{pmatrix} \epsilon_{11} \\ \epsilon_{12} \\ \epsilon_{21} \\ \epsilon_{22} \\ \epsilon_{31} \\ \epsilon_{32} \end{pmatrix} \tag{24.3}$$

$$\mathbf{y} = \mathbf{X} \ +$$

However,

$$\mathbf{X}'\mathbf{X} = \begin{pmatrix} \sum_i n_i & n_1 & n_2 & n_3 \\ n_1 & n_1 & 0 & 0 \\ n_2 & 0 & n_2 & 0 \\ n_3 & 0 & 0 & n_3 \end{pmatrix}$$

is **singular** thus does not exist, **b** is insolvable (infinite solutions)

Hence, we have to impose restrictions on the parameters to a model matrix **X** of full rank.

Whatever restriction we use, we still have:

$E(Y_{ij}) = \mu + \tau_i = \mu_i = mean\ response\ for\ i - th\ treatment$

24.5.2.1.1 Restriction on sum of tau $\sum_{i=1}^{a} \tau_i = 0$

implies

$$\mu = \mu + \frac{1}{a} \sum_{i=1}^{a} (\mu + \tau_i)$$

is the average of the treatment mean (grand mean) (overall mean)

$$\begin{aligned}\tau_i &= (\mu + \tau_i) - \mu = \mu_i - \mu \\ &= \text{treatment mean} - \text{grand mean} \\ &= \text{treatment effect}\end{aligned}$$

$$\tau_a = -\tau_1 - \tau_2 - ... - \tau_{a-1}$$

Hence, the mean for the a-th treatment is

$$\mu_a = \mu + \tau_a = \mu - \tau_1 - \tau_2 - ... - \tau_{a-1}$$

Hence, the model need only "a" parameters:

$$\mu, \tau_1, \tau_2, .., \tau_{a-1}$$

Equation (24.3) becomes

$$\begin{pmatrix} Y_{11} \\ Y_{12} \\ Y_{21} \\ Y_{22} \\ Y_{31} \\ Y_{32} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & -1 & -1 \\ 1 & -1 & -1 \end{pmatrix} \begin{pmatrix} \mu \\ \tau_1 \\ \tau_2 \end{pmatrix} + \begin{pmatrix} \epsilon_{11} \\ \epsilon_{12} \\ \epsilon_{21} \\ \epsilon_{22} \\ \epsilon_{31} \\ \epsilon_{32} \end{pmatrix} \tag{24.4}$$

$$\mathbf{y} = \mathbf{X} \ +$$

where $\beta \equiv [\mu, \tau_1, \tau_2]'$

Equation (24.2) with $\sum_i \tau_i = 0$ becomes

$$\begin{aligned}\mathbf{b} = \begin{bmatrix} \hat{\mu} \\ \hat{\tau}_1 \\ \hat{\tau}_2 \end{bmatrix} &= (\mathbf{x}'\mathbf{x})^{-1}\mathbf{x}'\mathbf{y} \\ &= \begin{bmatrix} \sum_i n_i & n_1 - n_3 & n_2 - n_3 \\ n_1 - n_3 & n_1 + n_3 & n_3 \\ n_2 - n_3 & n_3 & n_2 - n_3 \end{bmatrix}^{-1} \begin{bmatrix} Y_{..} \\ Y_{1.} - Y_{3.} \\ Y_{2.} - Y_{3.} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{3}\sum_{i=1}^{3} \bar{Y}_{i.} \\ \bar{Y}_{1.} - \frac{1}{3}\sum_{i=1}^{3} \bar{Y}_{i.} \\ \bar{Y}_{2.} - \frac{1}{3}\sum_{i=1}^{3} \bar{Y}_{i.} \end{bmatrix} \\ &= \begin{bmatrix} \hat{\mu} \\ \hat{\tau}_1 \\ \hat{\tau}_2 \end{bmatrix}\end{aligned}$$

and $\hat{\tau}_3 = -\hat{\tau}_1 - \hat{\tau}_2 = \bar{Y}_3 - \frac{1}{3}\sum_i \bar{Y}_{i.}$

24.5.2.1.2 Restriction on first tau In R, `lm()` uses the restriction $\tau_1 = 0$

For the previous example, for $n_1 = n_2 = n_3 = 2$, and $\tau_1 = 0$.

Then the treatment means can be written as:

$$\begin{aligned}
\mu_1 &= \mu + \tau_1 = \mu + 0 = \mu \\
\mu_2 &= \mu + \tau_2 \\
\mu_3 &= \mu + \tau_3
\end{aligned}$$

Hence, μ is the mean response for the first treatment

In the matrix form,

$$\begin{pmatrix} Y_{11} \\ Y_{12} \\ Y_{21} \\ Y_{22} \\ Y_{31} \\ Y_{32} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mu \\ \tau_2 \\ \tau_3 \end{pmatrix} + \begin{pmatrix} \epsilon_{11} \\ \epsilon_{12} \\ \epsilon_{21} \\ \epsilon_{22} \\ \epsilon_{31} \\ \epsilon_{32} \end{pmatrix}$$

$$\mathbf{y} = \mathbf{X} \ +$$

$\beta = [\mu, \tau_2, \tau_3]'$

$$\begin{aligned}
\mathbf{b} = \begin{bmatrix} \hat{\mu} \\ \hat{\tau}_2 \\ \hat{\tau}_3 \end{bmatrix} &= (\mathbf{x}'\mathbf{x})^{-1}\mathbf{x}'\mathbf{y} \\
&= \begin{bmatrix} \sum_i n_i & n_2 & n_3 \\ n_2 & n_2 & 0 \\ n_3 & 0 & n_3 \end{bmatrix}^{-1} \begin{bmatrix} Y_{..} \\ Y_{2.} \\ Y_{3.} \end{bmatrix} \\
&= \begin{bmatrix} \bar{Y_{1.}} \\ \bar{Y_{2.}} - \bar{Y_{1.}} \\ \bar{Y_{3.}} - \bar{Y_{1.}} \end{bmatrix}
\end{aligned}$$

$$E(\mathbf{b}) = \beta = \begin{bmatrix} \mu \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 - \mu_1 \\ \mu_3 - \mu_1 \end{bmatrix}$$

$$\begin{aligned}
var(\mathbf{b}) &= \sigma^2(\mathbf{X}'\mathbf{X})^{-1} \\
var(\hat{\mu}) &= var(\bar{Y_{1.}}) = \sigma^2/n_1 \\
var(\hat{\tau}_2) &= var(\bar{Y_{2.}} - \bar{Y_{1.}}) = \sigma^2/n_2 + \sigma^2/n_1 \\
var(\hat{\tau}_3) &= var(\bar{Y_{3.}} - \bar{Y_{1.}}) = \sigma^2/n_3 + \sigma^2/n_1
\end{aligned}$$

Note For all three parameterization, the ANOVA table is the same

- Model 1: $Y_{ij} = \mu_i + \epsilon_{ij}$
- Model 2: $Y_{ij} = \mu + \tau_i + \epsilon_{ij}$ where $\sum_i \tau_i = 0$
- Model 3: $Y_{ij} = \mu + \tau_i + \epsilon_{ij}$ where $\tau_1 = 0$

All models have the same calculation for $\hat{Y}$ as

$$\hat{\mathbf{Y}} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-\mathbf{1}}\mathbf{X}'\mathbf{Y} = \mathbf{PY} = \mathbf{Xb}$$

ANOVA Table

Source of Variation	SS	df	MS	F
Between Treatments	$\sum_i n_i(\bar{Y_{i.}} - \bar{Y_{..}})^2 = \mathbf{Y}'(\mathbf{P} - \mathbf{P_1})\mathbf{Y}$	a-1	$\frac{SSTR}{a-1}$	$\frac{MSTR}{MSE}$
Error (within treatments)	$\sum_i \sum_j (Y_{ij} - \bar{Y_{i.}})^2 = \mathbf{e}'\mathbf{e}$	N-a	$\frac{SSE}{N-a}$	
Total (corrected)	$\sum_i n_i(\bar{Y_{i.}} - \bar{Y_{..}})^2 = \mathbf{Y}'\mathbf{Y} - \mathbf{Y}'\mathbf{P_1}\mathbf{Y}$	N-1		

where $\mathbf{P_1} = \frac{1}{n}\mathbf{J}$

The F-statistic here has $(a-1, N-a)$ degrees of freedom, which gives the same value for all three parameterization, but the hypothesis test is written a bit different:

$$H_0 : \mu_1 = \mu_2 = ... = \mu_a$$
$$H_0 : \mu + \tau_1 = \mu + \tau_2 = ... = \mu + \tau_a$$
$$H_0 : \tau_1 = \tau_2 = ... = \tau_a$$

The F-test here serves as a preliminary analysis, to see if there is any difference at different factors. For more in-depth analysis, we consider different testing of treatment effects.

24.5.2.2 Testing of Treatment Effects

- A Single Treatment Mean μ_i
- A Differences Between Treatment Means
- A Contrast Among Treatment Means
- A Linear Combination of Treatment Means

24.5.2.2.1 Single Treatment Mean We have $\hat{\mu_i} = \bar{Y_{i.}}$ where

- $E(\bar{Y_{i.}}) = \mu_i$
- $var(\bar{Y_i}) = \sigma^2/n_i$ estimated by $s^2(\bar{Y_{i.}}) = MSE/n_i$

Since $\frac{\bar{Y_{i.}}-\mu_i}{s(\bar{Y_{i.}})} \sim t_{N-a}$ and the confidence interval for μ_i is $\bar{Y_{i.}} \pm t_{1-\alpha/2;N-a} s(\bar{Y_{i.}})$, then we can do a t-test for the means difference with some constant c

$$H_0 : \mu_i = c$$
$$H_1 : \mu_i \neq c$$

where

$$T = \frac{\bar{Y_{i.}} - c}{s(\bar{Y_{i.}})}$$

follows t_{N-a} when H_0 is true.
If $|T| > t_{1-\alpha/2;N-a}$, we can reject H_0

24.5.2.2.2 Differences Between Treatment Means Let $D = \mu_i - \mu_i'$, also known as **pairwise comparison**
D can be estimated by $\hat{D} = \bar{Y_i} - \bar{Y_i}'$ is unbiased ($E(\hat{D}) = \mu_i - \mu_i'$)

Since $\bar{Y_i}$ and $\bar{Y_i}'$ are independent, then

$$var(\hat{D}) = var(\bar{Y_i}) + var(\bar{Y_{i'}}) = \sigma^2(1/n_i + 1/n_i')$$

can be estimated with

$$s^2(\hat{D}) = MSE(1/n_i + 1/n_i')$$

With the single treatment inference,

$$\frac{\hat{D} - D}{s(\hat{D})} \sim t_{N-a}$$

hence,

$$\hat{D} \pm t_{(1-\alpha/2;N-a)} s(\hat{D})$$

Hypothesis tests:

$$H_0 : \mu_i = \mu_i'$$
$$H_a : \mu_i \neq \mu_i'$$

can be tested by the following statistic

$$T = \frac{\hat{D}}{s(\hat{D})} \sim t_{1-\alpha/2;N-a}$$

reject H_0 if $|T| > t_{1-\alpha/2;N-a}$

24.5.2.2.3 Contrast Among Treatment Means generalize the comparison of two means, we have **contrasts**

A contrast is a linear combination of treatment means:

$$L = \sum_{i=1}^{a} c_i \mu_i$$

where each c_i is non-random constant and sum to 0:

$$\sum_{i=1}^{a} c_i = 0$$

An unbiased estimator of a contrast L is

$$\hat{L} = \sum_{i=1}^{a} c_i \bar{Y}_{i.}$$

and $E(\hat{L}) = L$. Since the $\bar{Y}_{i.}$, i = 1,..., a are independent.

$$\begin{aligned} var(\hat{L}) &= var(\sum_{i=1}^{a} c_i \bar{Y}_{i.}) = \sum_{i=1}^{a} var(c_i \bar{Y}_i) \\ &= \sum_{i=1}^{a} c_i^2 var(\bar{Y}_i) = \sum_{i=1}^{a} c_i^2 \sigma^2 / n_i \\ &= \sigma^2 \sum_{i=1}^{a} c_i^2 / n_i \end{aligned}$$

Estimation of the variance:

$$s^2(\hat{L}) = MSE \sum_{i=1}^{a} \frac{c_i^2}{n_i}$$

$\hat{L}$ is normally distributed (since it is a linear combination of independent normal random variables).

Then, since SSE/σ^2 is χ^2_{N-a}

$$\frac{\hat{L} - L}{s(\hat{L})} \sim t_{N-a}$$

A $1-\alpha$ confidence limits are given by

$$\hat{L} \pm t_{1-\alpha/2;N-a} s(\hat{L})$$

Hypothesis testing

$$H_0 : L = 0$$
$$H_a : L \neq 0$$

with

$$T = \frac{\hat{L}}{s(\hat{L})}$$

reject H_0 if $|T| > t_{1-\alpha/2;N-a}$

24.5.2.2.4 Linear Combination of Treatment Means just like contrast $L = \sum_{i=1}^{a} c_i \mu_i$ but no restrictions on the c_i coefficients.

Tests on a single treatment mean, two treatment means, and contrasts can all be considered form the same perspective.

$$H_0 : \sum c_i \mu_i = c$$
$$H_a : \sum c_i \mu_i \neq c$$

The test statistics (t-stat) can be considered equivalently as F-tests; $F = (T)^2$ where $F \sim F_{1,N-a}$. Since the numerator degrees of freedom is always 1 in these cases, we refer to them as single-degree-of-freedom tests.

Multiple Contrasts

To test simultaneously $k \geq 2$ contrasts, let $T_1, ..., T_k$ be the t-stat. The joint distribution of these random variables is a multivariate t-distribution (the tests are dependent since they re based on the same data).

Limitations for comparing multiple contrasts:

1. The confidence coefficient $1-\alpha$ only applies to a particular estimate, not a series of estimates; similarly, the Type I error rate, α, applies to a particular test, not a series of tests. Example: 3 t-tests at $\alpha = 0.05$, if tests are independent (which they are not), $0.95^3 = 0.857$ (thus $\alpha - 0.143$ not 0.05)

2. The confidence coefficient $1-\alpha$ and significance level α are appropriate only if the test was not suggest by the data.

   - often, the results of an experiment suggest important (i.e.,..g, potential significant) relationships.
   - the process of studying effects suggests by the data is called **data snooping**

Multiple Comparison Procedures:

- Tukey
- Scheffe
- Bonferroni

24.5.2.2.4.1 Tukey All pairwise comparisons of factor level means. All pairs $D = \mu_i - \mu_i'$ or all tests of the form:

$$H_0 : \mu_i - \mu_i' = 0$$
$$H_a : \mu_i - \mu_i' \neq 0$$

- When all sample sizes are equal ($n_1 = n_2 = ... = n_a$) then the Tukey method family confidence coefficient is exactly $1-\alpha$ and the significance level is exactly α

- When the sample sizes are not equal, the family confidence coefficient is greater than $1-\alpha$ (i.e., the significance level is less than α) so the test **conservative**

- Tukey considers the **studentized range distribution**. If we have $Y_1, .., Y_r$, observations from a normal distribution with mean α and variance σ^2. Define:

  $$w = max(Y_i) - min(Y_i)$$

  as the range of the observations. Let s^2 be an estimate of σ^2 with v degrees of freedom. Then,

  $$q(r, v) = \frac{w}{s}$$

  is called the studentized range. The distribution of q uses a special table.

Notes

- when we are not interested in testing all pairwise comparison,s the confidence coefficient for the family of comparisons under consideration will be greater than $1-\alpha$ (with the significance level less than α)
- Tukey can be used for "data snooping" as long as the effects to be studied on the basis of preliminary data analysis are pairwise comparisons.

24.5.2.2.4.2 Scheffe This method applies when the family of interest is the set of possible contrasts among the treatment means:

$$L = \sum_{i=1}^{a} c_i \mu_i$$

where $\sum_{i=1}^{a} c_i = 0$

That is, the family of all possible contrasts L or

$$H_0 : L = 0$$
$$H_a : L \neq 0$$

The family confidence level for the Scheffe procedure is exactly $1-\alpha$ (i.e., significance level $= \alpha$) whether the sample sizes are equal or not.

For simultaneous confidence intervals,

$$\hat{L} \pm Ss(\hat{L})$$

where $\hat{L} = \sum c_i \bar{Y}_{i.}$, $s^2(\hat{L}) = MSE \sum c_i^2/n_i$ and $S^2 = (a-1)f_{1-\alpha;a-1,N-a}$

The Scheffe procedure considers

$$F = \frac{\hat{L}^2}{(a-1)s^2(\hat{L})}$$

where we reject H_0 at the family significance level α if $F > f_{(1-\alpha;a-1,N-a)}$

Note

- Since applications of the Scheffe never involve all conceivable contrasts, the **finite family** confidence coefficient will be larger than $1-\alpha$, so $1-\alpha$ is a lower bound. Thus, people often consider a larger α (e.g., 90% confidence interval)
- Scheffe can be used for "data scooping" since the family of statements contains all possible contrasts.
- If only pairwise comparisons are to be considered, The Tukey procedure gives narrower confidence limits.

24.5.2.2.4.3 Bonferroni Applicable whether the sample sizes are equal or unequal.

For the confidence intervals,

$$\hat{L} \pm Bs(\hat{L})$$

where $B = t_{(1-\alpha/(2g);N-a)}$ and g is the number of comparisons in the family.

Hypothesis testing

$$H_0 : L = 0$$
$$H_a : L \neq 0$$

Let $T = \frac{\hat{L}}{s(\hat{L})}$ and reject H_0 if $|T| > t_{1-\alpha/(2g),N-a}$

Notes

- If all pairwise comparisons are of interest, the Tukey procedure is superior (narrower confidence intervals). If not, Bonferroni may be better.
- Bonferroni is better than Scheffe when the number of contrasts is about the same as the treatment levels (or less).
- Recommendation: compute all threes and pick the smallest.
- Bonferroni can't be used for **data snooping**

24.5.2.2.4.4 Fisher's LSD does not control for family error rate

use t-stat for testing

$$H_0 : \mu_i = \mu_j$$

t-stat

$$t = \frac{\bar{y}_i - \bar{y}_j}{\sqrt{MSE(\frac{1}{n_i} + \frac{1}{n_j})}}$$

24.5.2.2.4.5 Newman-Keuls Do not recommend using this test since it has less power than ANOVA.

24.5.2.2.5 Multiple comparisons with a control

24.5.2.2.5.1 Dunnett We have a groups where the last group is the control group, and the $a-1$ treatment groups.

Then, we compare treatment groups to the control group. Hence, we have $a-1$ contrasts (i.e., $a-1$ pairwise comparisons)

24.5.2.2.6 Summary When choosing a multiple contrast method:

- Pairwise
  - Equal groups sizes: Tukey
  - Unequal groups sizes: Tukey, Scheffe
- Not pairwise
  - with control: Dunnett
  - general: Bonferroni, Scheffe

24.5.3 Single Factor Random Effects Model

Also known as ANOVA Type II models.

Treatments are chosen at from from larger population. We extend inference to all treatments in the population and not restrict our inference to those treatments that happened to be selected for the study.

24.5.3.1 Random Cell Means

$$Y_{ij} = \mu_i + \epsilon_{ij}$$

where

- $\mu_i \sim N(\mu, \sigma_\mu^2)$ and independent
- $\epsilon_{ij} \sim N(0, \sigma^2)$ and independent

μ_i and ϵ_{ij} are mutually independent for $i = 1, ..., a; j = 1, ..., n$

With all treatment sample sizes are equal

$$\begin{aligned} E(Y_{ij}) &= E(\mu_i) = \mu \\ var(Y_{ij}) &= var(\mu_i) + var(\epsilon_i) = \sigma_\mu^2 + \sigma^2 \end{aligned}$$

Since Y_{ij} are not independent

$$\begin{aligned} cov(Y_{ij}, Y_{ij'}) &= E(Y_{ij}Y_{ij'}) - E(Y_{ij})E(Y_{ij'}) \\ &= E(\mu_i^2 + \mu_i\epsilon_{ij'} + \mu_i\epsilon_{ij} + \epsilon_{ij}\epsilon_{ij'}) - \mu^2 \\ &= \sigma_\mu^2 + \mu^2 - \mu^2 && \text{if} j \neq j' \\ &= \sigma_\mu^2 && \text{if} j \neq j' \end{aligned}$$

$$\begin{aligned} cov(Y_{ij}, Y_{i'j'}) &= E(\mu_i\mu_{i'} + \mu_i\epsilon_{i'j'} + \mu_{i'}\epsilon_{ij} + \epsilon_{ij}\epsilon_{i'j'}) - \mu^2 \\ &= \mu^2 - \mu^2 && \text{if } i \neq i' \\ &= 0 \end{aligned}$$

Hence,

- all observations have the same variance
- any two observations from the same treatment have covariance σ_μ^2
- The correlation between any two responses from the same treatment:

$$\rho(Y_{ij}, Y_{ij'}) = \frac{\sigma_\mu^2}{\sigma_\mu^2 + \sigma^2} \quad j \neq j'$$

Inference

Intraclass Correlation Coefficient

$$\frac{\sigma_\mu^2}{\sigma^2 + \sigma_\mu^2}$$

which measures the proportion of total variability of Y_{ij} accounted for by the variance of μ_i

$$H_0 : \sigma_\mu^2 = 0$$
$$H_a : \sigma_\mu^2 \neq 0$$

H_0 implies $\mu_i = \mu$ for all i, which can be tested by the F-test in ANOVA.

The understandings of the Single Factor Fixed Effects Model and the Single Factor Random Effects Model are different, the ANOVA is same for the one factor model. The difference is in the expected mean squares

Random Effects Model	**Fixed Effects** Model
$E(MSE) = \sigma^2$	$E(MSE) = \sigma^2$
$E(MSTR) = \sigma^2 - n\sigma_\mu^2$	$E(MSTR) = \sigma^2 + \frac{\sum_i n_i(\mu_i - \mu)^2}{a-1}$

If σ_μ^2, then MSE and MSTR have the same expectation (σ^2). Otherwise, $E(MSTR) > E(MSE)$. Large values of the statistic

$$F = \frac{MSTR}{MSE}$$

suggest we reject H_0.

Since $F \sim F_{(a-1,a(n-1))}$ when H_0 holds. If $F > f_{(1-\alpha;a-1,a(n-1))}$ we reject H_0.

If sample sizes are not equal, F-test can still be used, but the df are $a-1$ and $N-a$.

24.5.3.1.1 Estimation of μ An unbiased estimator of $E(Y_{ij}) = \mu$ is the grand mean: $\hat{\mu} = \hat{Y}_{..}$

The variance of this estimator is

$$\begin{aligned} var(\bar{Y}_{..}) &= var(\sum_i \bar{Y}_{i.}/a) \\ &= \frac{1}{a^2}\sum_i var(\bar{Y}_{i.}) \\ &= \frac{1}{a^2}\sum_i (\sigma_\mu^2 + \sigma^2/n) \\ &= \frac{1}{a^2}(\sigma_\mu^2 + \sigma^2/n) \\ &= \frac{n\sigma_\mu^2 + \sigma^2}{an} \end{aligned}$$

An unbiased estimator of this variance is $s^2(\bar{Y}) = \frac{MSTR}{an}$. Thus $\frac{\bar{Y}_{..}-\mu}{s(\bar{Y}_{..})} \sim t_{a-1}$

A $1-\alpha$ confidence interval is $\bar{Y}_{..} \pm t_{(1-\alpha/2;a-1)}s(\bar{Y}_{..})$

24.5.3.1.2 Estimation of $\sigma_\mu^2/(\sigma_\mu^2 + \sigma^2)$ In the random and fixed effects model, MSTR and MSE are independent. When the sample sizes are equal ($n_i = n$ for all i),

$$\frac{\frac{MSTR}{n\sigma_\mu^2+\sigma^2}}{\frac{MSE}{\sigma^2}} \sim f_{(a-1,a(n-1))}$$

$$P(f_{(\alpha/2;a-1,a(n-1))} \leq \frac{\frac{MSTR}{n\sigma_\mu^2+\sigma^2}}{\frac{MSE}{\sigma^2}} \leq f_{(1-\alpha/2;a-1,a(n-1))}) = 1-\alpha$$

$$\begin{aligned} L &= \frac{1}{n}(\frac{MSTR}{MSE}(\frac{1}{f_{(1-\alpha/2;a-1,a(n-1))}}) - 1) \\ U &= \frac{1}{n}(\frac{MSTR}{MSE}(\frac{1}{f_{(\alpha/2;a-1,a(n-1))}}) - 1) \end{aligned}$$

The lower and upper (L^*, U^*) confidence limits for $\frac{\sigma_\mu^2}{\sigma_\mu^2+\sigma^2}$

$$\begin{aligned} L^* &= \frac{L}{1+L} \\ U^* &= \frac{U}{1+U} \end{aligned}$$

If the lower limit for $\frac{\sigma_\mu^2}{\sigma^2}$ is negative, it is customary to set $L = 0$.

24.5.3.1.3 Estimation of σ^2 $a(n-1)MSE/\sigma^2 \sim \chi^2_{a(n-1)}$, the $(1-\alpha)$ confidence interval for σ^2:

$$\frac{a(n-1)MSE}{\chi^2_{1-\alpha/2;a(n-1)}} \leq \sigma^2 \leq \frac{a(n-1)MSE}{\chi^2_{\alpha/2;a(n-1)}}$$

can also be used in case sample sizes are not equal - then df is N-a.

24.5.3.1.4 Estimation of σ^2_μ $E(MSE) = \sigma^2$ $E(MSTR) = \sigma^2 + n\sigma^2_\mu$. Hence,

$$\sigma^2_\mu = \frac{E(MSTR) - E(MSE)}{n}$$

An unbiased estimator of σ^2_μ is given by

$$s^2_\mu = \frac{MSTR - MSE}{n}$$

if $s^2_\mu < 0$, set $s^2_\mu = 0$

If sample sizes are not equal,

$$s^2_\mu = \frac{MSTR - MSE}{n'}$$

where $n' = \frac{1}{a-1}(\sum_i n_i - \frac{\sum_i n_i^2}{\sum_i n_i})$

no exact confidence intervals for σ^2_μ, but we can approximate intervals.

Satterthewaite Procedure can be used to construct approximate confidence intervals for linear combination of expected mean squares
A linear combination:

$$\sigma^2_\mu = \frac{1}{n}E(MSTR) + (-\frac{1}{n})E(MSE)$$

$$S = d_1E(MS_1) + .. + d_hE(MS_h)$$

where d_i are coefficients.

An unbiased estimator of S is

$$\hat{S} = d_1MS_1 + ... + d_hMS_h$$

Let df_i be the degrees of freedom associated with the mean square MS_i. The **Satterthwaite** approximation:

$$\frac{(df)\hat{S}}{S} \sim \chi^2_{df}$$

where

$$df = \frac{(d_1 MS_1 + ... + d_h MS_h)^2}{(d_1 MS_1)^2/df_1 + ... + (d_h MS_h)^2/df_h}$$

An approximate $1-\alpha$ confidence interval for S:

$$\frac{(df)\hat{S}}{\chi^2_{1-\alpha/2;df}} \leq S \leq \frac{(df)\hat{S}}{\chi^2_{\alpha/2;df}}$$

For the single factor random effects model

$$\frac{(df)s^2_\mu}{\chi^2_{1-\alpha/2;df}} \leq \sigma^2_\mu \leq \frac{(df)s^2_\mu}{\chi^2_{\alpha/2;df}}$$

where

$$df = \frac{(sn^2_\mu)^2}{\frac{(MSTR)^2}{a-1} + \frac{(MSE)^2}{a(n-1)}}$$

24.5.3.2 Random Treatment Effects Model

$$\tau_i = \mu_i - E(\mu_i) = \mu_i - \mu$$

we have $\mu_i = \mu + \tau_i$ and

$$Y_{ij} = \mu + \tau_i + \epsilon_{ij}$$

where

- μ = constant, common to all observations
- $\tau_i \sim N(0, \sigma^2_\tau)$ independent (random variables)
- $\epsilon_{ij} \sim N(0, \sigma^2)$ independent.
- τ_i, ϵ_{ij} are independent (i=1,...,a; j =1,..,n)
- our model is concerned with only balanced single factor ANOVA.

Diagnostics Measures

- Non-constant error variance (plots, Levene test, Hartley test).

- Non-independence of errors (plots, Durban-Watson test).
- Outliers (plots, regression methods).
- Non-normality of error terms (plots, Shapiro-Wilk, Anderson-Darling).
- Omitted Variable Bias (plots)

Remedial

- Weighted Least Squares
- [Transformations]
- Non-parametric Procedures.

Note

- Fixed effect ANOVA is relatively robust to
  - non-normality
  - unequal variances when sample sizes are approximately equal; at least the F-test and multiple comparisons. However, single comparisons of treatment means are sensitive to unequal variances.
- Lack of independence can seriously affect both fixed and random effect ANVOA.

24.5.4 Two Factor Fixed Effect ANOVA

The multi-factor experiment is

- more efficient
- provides more info
- gives more validity to the findings.

24.5.4.1 Balanced

Assumption:

- All treatment sample sizes are equal
- All treatment means are of equal importance

Assume:

- Factor A has `a` levels and Factor B has `b` levels. All $a \times b$ factor levels are considered.
- The number of treatments for each level is n. $N = abn$ observations in the study.

24.5.4.1.1 Cell Means Model

$$Y_{ijk} = \mu_{ij} + \epsilon_{ijk}$$

where

- μ_{ij} are fixed parameters (cell means)
- $i = 1, ..., a =$ the levels of Factor A
- $j = 1, ..., b =$ the levels of Factor B.
- $\epsilon_{ijk} \sim$ indep $N(0, \sigma^2)$ for $i = 1, ..., a$, $j = 1, .., b$ and $k = 1, .., n$

And

$$E(Y_{ijk}) = \mu_{ij}$$
$$var(Y_{ijk}) = var(\epsilon_{ijk}) = \sigma^2$$

Hence,

$$Y_{ijk} \sim \text{indep } N(\mu_{ij}, \sigma^2)$$

And the model is

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon$$

Thus,

$$E(\mathbf{Y}) = \mathbf{X}\beta$$
$$var(\mathbf{Y}) = \sigma^2\mathbf{I}$$

Interaction

$$(\alpha\beta)_{ij} = \mu_{ij} - (\mu_{..} + \alpha_i + \beta_j)$$

where

- $\mu_{..} = \sum_i \sum_j \mu_{ij}/ab$ is the grand mean
- $\alpha_i = \mu_{i.} - \mu_{..}$ is the main effect for factor A at the i-th level
- $\beta_j = \mu_{.j} - \mu_{..}$ is the main effect for factor B at the j-th level
- $(\alpha\beta)_{ij}$ is the interaction effect when factor A is at the i-th level and factor B is at the j-th level.
- $(\alpha\beta)_{ij} = \mu_{ij} - \mu_{i.} - \mu_{.j} + \mu_{..}$

Examine interactions:

- Examine whether all μ_{ij} can be expressed as the sums $\mu_{..} + \alpha_i + \beta_j$
- Examine whether the difference between the mean responses for any two levels of factor B is the same for all levels of factor A.
- Examine whether the difference between the mean response for any two levels of factor A is the same for all levels of factor B

- Examine whether the treatment mean curves for the different factor levels in a treatment plot are parallel.

For $j = 1, ..., b$

$$\begin{aligned}\sum_i (\alpha\beta)_{ij} &= \sum_i (\mu_{ij} - \mu_{..} - \alpha_i - \beta_j) \\ &= \sum_i \mu_{ij} - a\mu_{..} - \sum_i \alpha_i - a\beta_j \\ &= a\mu_{.j} - a\mu_{..} - \sum_i (\mu_{i.} - \mu_{..}) - a(\mu_{.j} - \mu_{..}) \\ &= a\mu_{.j} - a\mu_{..} - a\mu_{..} + a\mu_{..} - a(\mu_{.j} - \mu_{..}) \\ &= 0\end{aligned}$$

Similarly, $\sum_j (\alpha\beta) = 0, i = 1, ..., a$ and $\sum_i \sum_j (\alpha\beta)_{ij} = 0$, $\sum_i \alpha_i = 0$, $\sum_j \beta_j = 0$

24.5.4.1.2 Factor Effects Model

$$\begin{aligned}\mu_{ij} &= \mu_{..} + \alpha_i + \beta_j + (\alpha\beta)_{ij} \\ Y_{ijk} &= \mu_{..} + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \epsilon_{ijk}\end{aligned}$$

where

- $\mu_{..}$ is a constant
- α_i are constants subject to the restriction $\sum_i \alpha_i = 0$
- β_j are constants subject to the restriction $\sum_j \beta_j = 0$
- $(\alpha\beta)_{ij}$ are constants subject to the restriction $\sum_i (\alpha\beta)_{ij} = 0$ for $j = 1, ..., b$ and $\sum_j (\alpha\beta)_{ij} = 0$ for $i = 1, ..., a$
- $\epsilon_{ijk} \sim$ indep $N(0, \sigma^2)$ for $k = 1, .., n$

We have

$$\begin{aligned}E(Y_{ijk}) &= \mu_{..} + \alpha_i + \beta_j + (\alpha\beta)_{ij} \\ var(Y_{ijk}) &= \sigma^2 \\ Y_{ijk} &\sim N(\mu_{..} + \alpha_i + \beta_j + (\alpha\beta)_{ij}, \sigma^2)\end{aligned}$$

We have $1 + a + b + ab$ parameters. But there are ab parameters in the Cell Means Model. In the Factor Effects Model, the restrictions limit the number of parameters that can be estimated:

$$\begin{aligned}1 &\text{ for } \mu_{..} \\ (a-1) &\text{ for } \alpha_i \\ (b-1) &\text{ for } \beta_j \\ (a-1)(b-1) &\text{ for } (\alpha\beta)_{ij}\end{aligned}$$

Hence, there are

$$1 + a - 1 + b - 1 + ab - a - b + 1 = ab$$

parameters in the model.

We can have several restrictions when considering the model in the form $\mathbf{Y} = \mathbf{X}\beta + \epsilon$

One way:

$$\begin{aligned}
\alpha_a &= \alpha_1 - \alpha_2 - ... - \alpha_{a-1} \\
\beta_b &= -\beta_1 - \beta_2 - ... - \beta_{b-1} \\
(\alpha\beta)_{ib} &= -(\alpha\beta)_{i1} - (\alpha\beta)_{i2} - ... - (\alpha\beta)_{i,b-1}; i = 1, .., a \\
(\alpha\beta)_{aj} &= -(\alpha\beta)_{1j} - (\alpha\beta)_{2j} - ... - (\alpha\beta)_{a-1,j}; j = 1, .., b
\end{aligned}$$

We can fit the model by least squares or maximum likelihood

Cell Means Model
minimize

$$Q = \sum_i \sum_j \sum_k (Y_{ijk} - \mu_{ij})^2$$

estimators

$$\begin{aligned}
\hat{\mu}_{ij} &= \bar{Y}_{ij} \\
\hat{Y}_{ijk} &= \bar{Y}_{ij} \\
e_{ijk} = Y_{ijk} - \hat{Y}_{ijk} &= Y_{ijk} - \bar{Y}_{ij}
\end{aligned}$$

Factor Effects Model

$$Q = \sum_i \sum_j \sum_k (Y_{ijk} - \mu_{..} - \alpha_i = \beta_j - (\alpha\beta)_{ij})^2$$

subject to the restrictions

$$\sum_i \alpha_i = 0$$
$$\sum_j \beta_j = 0$$
$$\sum_i (\alpha\beta)_{ij} = 0$$
$$\sum_j (\alpha\beta)_{ij} = 0$$

estimators

$$\begin{aligned}
\hat{\mu}_{..} &= \bar{Y}_{...} \\
\hat{\alpha}_i &= \bar{Y}_{i..} - \bar{Y}_{...} \\
\hat{\beta}_j &= \bar{Y}_{.j.} - \bar{Y}_{...} \\
(\hat{\alpha\beta})_{ij} &= \bar{Y}_{ij.} - \bar{Y}_{i..} - \bar{Y}_{.j.} + \bar{Y}_{...}
\end{aligned}$$

The fitted values

$$\hat{Y}_{ijk} = \bar{Y}_{...} + (\bar{Y}_{i..} - \bar{Y}_{...}) + (\bar{Y}_{.j.} - \bar{Y}_{...}) + (\bar{Y}_{ij.} - \bar{Y}_{i..} - \bar{Y}_{.j.} + \bar{Y}_{...}) = \bar{Y}_{ij.}$$

where

$$\begin{aligned}
e_{ijk} &= Y_{ijk} - \bar{Y}_{ij.} \\
e_{ijk} &\sim \text{ indep } (0, \sigma^2)
\end{aligned}$$

and

$$\begin{aligned}
s^2_{\hat{\mu}_{..}} &= \frac{MSE}{nab} \\
s^2_{\hat{\alpha}_i} &= MSE(\frac{1}{nb} - \frac{1}{nab}) \\
s^2_{\hat{\beta}_j} &= MSE(\frac{1}{na} - \frac{1}{nab}) \\
s^2_{(\hat{\alpha\beta})_{ij}} &= MSE(\frac{1}{n} - \frac{1}{na} - \frac{1}{nb} + \frac{1}{nab})
\end{aligned}$$

24.5.4.1.2.1 Partitioning the Total Sum of Squares

$$Y_{ijk} - \bar{Y}_{...} = \bar{Y}_{ij.} - \bar{Y}_{...} + Y_{ijk} - \bar{Y}_{ij.}$$

$Y_{ijk} - \bar{Y}_{...}$: Total deviation
$\bar{Y}_{ij.} - \bar{Y}_{...}$: Deviation of treatment mean from overall mean
$Y_{ijk} - \bar{Y}_{ij.}$: Deviation of observation around treatment mean (residual).

$$\sum_i \sum_j \sum_k (Y_{ijk} - \bar{Y}_{...})^2 = n \sum_i \sum_j (\bar{Y}_{ij.} - \bar{Y}_{...})^2 + \sum_i \sum_j sum_k (Y_{ijk} - \bar{ij.})^2$$
$$SSTO = SSTR + SSE$$

(cross product terms are 0)

$$\bar{Y}_{ij.} - \bar{Y}_{...} = \bar{Y}_{i..} - \bar{Y}_{...} + \bar{Y}_{.j.} - \bar{Y}_{...} + \bar{Y}_{ij.} - \bar{Y}_{i..} - \bar{Y}_{.j.} + \bar{Y}_{...}$$

squaring and summing:

$$n \sum_i \sum_j (\bar{Y}_{ij.} - \bar{Y}_{...})^2 = nb \sum_i (\bar{Y}_{i..} - \bar{Y}_{...})^2 + na \sum_j (\bar{Y}_{.j.} - \bar{Y}_{...})^2$$
$$+ n \sum_i \sum_j (\bar{Y}_{ij.} - \bar{Y}_{i..} - \bar{Y}_{.j.} + \bar{Y}_{...})^2$$
$$SSTR = SSA + SSB + SSAB$$

The interaction term from

$$SSAB = SSTO - SSE - SSA - SSB$$
$$SSAB = SSTR - SSA - SSB$$

where

- SSA is the factor A sum of squares (measures the variability of the estimated factor A level means $\bar{Y}_{i..}$)- the more variable, the larger SSA
- SSB is the factor B sum of squares
- $SSAB$ is the interaction sum of squares, measuring the variability of the estimated interactions.

24.5.4.1.2.2 Partitioning the df $N = abn$ cases and ab treatments.

For one-way ANOVA and regression, the partition has df:

$$SS : SSTO = SSTR + SSE$$

$$df : N - 1 = (ab - 1) + (N - ab)$$

we must further partition the $ab-1$ df with SSTR

$$SSTR = SSA + SSB + SSAB$$

$$ab-1 = (a-1)+(b-1)+(a-1)(b-1)$$

- $df_{SSA} = a-1$: a treatment deviations but 1 df is lost due to the restriction $\sum(\bar{Y}_{i..} - \bar{Y}_{...}) = 0$
- $df_{SSB} = b-1$: b treatment deviations but 1 df is lost due to the restriction $\sum(\bar{Y}_{.j.} - \bar{Y}_{...}) = 0$
- $df_{SSAB} = (a-1)(b-1) = (ab-1)-(a-1)-(b-1)$: ab interactions, there are (a+b-1) restrictions, so df = ab-a-(b-1)= (a-1)(b-1)

24.5.4.1.2.3 Mean Squares

$$\begin{aligned} MSA &= \frac{SSA}{a-1} \\ MSB &= \frac{SSB}{b-1} \\ MSAB &= \frac{SSAB}{(a-1)(b-1)} \end{aligned}$$

The expected mean squares are

$$\begin{aligned} E(MSE) &= \sigma^2 \\ E(MSA) &= \sigma^2 + nb\frac{\sum \alpha_i^2}{a-1} = \sigma^2 + nb\frac{\sum(\sum_{i.} - \mu_{..})^2}{a-1} \\ E(MSB) &= \sigma^2 + na\frac{\sum \beta_i^2}{b-1} = \sigma^2 + na\frac{\sum(\sum_{.j} - \mu_{..})^2}{b-1} \\ E(MSAB) &= \sigma^2 + n\frac{\sum\sum(\alpha\beta)_{ij}^2}{(a-1)(b-1)} = \sigma^2 + n\frac{\sum(\mu_{ij} - \mu_{i.} - \mu_{.j} + \mu_{..})^2}{(a-1)(b-1)} \end{aligned}$$

If there are no factor A main effects (all $\mu_{i.} = 0$ or $\alpha_i = 0$) the MSA and MSE have the same expectation; otherwise MSA > MSE. Same for factor B, and interaction effects. which case we can examine F-statistics.

Interaction

$$\begin{aligned} H_0 &: \mu_{ij} - \mu_{i.} - \mu_{.j} + \mu_{..} = 0 \qquad \text{for all i,j} \\ H_a &: \mu_{ij} - \mu_{i.} - \mu_{.j} + \mu_{..} \neq 0 \quad \text{for some i,j} \end{aligned}$$

or

$$H_0 : \text{All}(\alpha\beta)_{ij} = 0$$
$$H_a : \text{Not all}(\alpha\beta) = 0$$

Let $F = \frac{MSAB}{MSE}$. When H_0 is true $F \sim f_{((a-1)(b-1),ab(n-1))}$. So reject H_0 when $F > f_{((a-1)(b-1),ab(n-1))}$

Factor A main effects:

$$H_0 : \mu_{1.} = \mu_{2.} = ... = \mu_{a.}$$
$$H_a : \text{Not all } \mu_{i.} \text{ are equal}$$

or

$$H_0 : \alpha_1 = ... = \alpha_a = 0$$
$$H_a : \text{Not all } \alpha_i \text{ are equal to } 0$$

$F = \frac{MSA}{MSE}$ and reject H_0 if $F > f_{(1-\alpha;a-1,ab(n-1))}$

24.5.4.1.2.4 Two-way ANOVA

Source of Variation	SS	df	MS	F
Factor A	SSA	$a-1$	$MSA = SSA/(a-1)$	MSA/MSE
Factor B	SSB	$b-1$	$MSB = SSB/(b-1)$	MSB/MSE
AB interactions	$SSAB$	$(a-1)(b-1)$	$MSAB = SSAB/MSE$	
Error	SSE	$ab(n-1)$	$MSE = SSE/ab(n-1)$	
Total (corrected)	$SSTO$	$abn-1$		

Doing 2-way ANOVA means you always check interaction first, because if there are significant interactions, checking the significance of the main effects becomes moot.

The main effects concern the mean responses for levels of one factor averaged over the levels of the other factor. When interaction is present, we can't conclude that a given factor has no effect, even if these averages are the same. It means that the effect of the factor depends on the level of the other factor.

On the other hand, if you can establish that there is no interaction, then you can consider inference on the factor main effects, which are then said to be

additive.
And we can also compare factor means like the Single Factor Fixed Effects Model using Tukey, Scheffe, Bonferroni.

We can also consider contrasts in the 2-way model

$$L = \sum c_i \mu_i$$

where $\sum c_i = 0$
which is estimated by

$$\hat{L} = \sum c_i \bar{Y}_{i..}$$

with variance

$$\sigma^2(\hat{L}) = \frac{\sigma^2}{bn} \sum c_i^2$$

and variance estimate

$$\frac{MSE}{bn} \sum c_i^2$$

Orthogonal Contrasts

$$L_1 = \sum c_i \mu_i, \sum c_i = 0$$
$$L_2 = \sum d_i \mu_i, \sum d_i = 0$$

these contrasts are said to be **orthogonal** if

$$\sum \frac{c_i d_i}{n_i} = 0$$

in balanced case $\sum c_i d_i = 0$

$$\begin{aligned} cov(\hat{L}_1, \hat{L}_2) &= cov(\sum_i c_i \bar{Y}_{i..}, \sum_l d_l \bar{Y}_{l..}) \\ &= \sum_i \sum_l c_i d_l cov(\bar{Y}_{i..}, \bar{Y}_{l..}) \\ &= \sum_i c_i d_i \frac{\sigma^2}{bn} = 0 \end{aligned}$$

Orthogonal contrasts can be used to further partition the model sum of squares. There are many sets of orthogonal contrasts and thus, many ways to partition the sum of squares.

A special set of orthogonal contrasts that are used when the levels of a factor can be assigned values on a metric scale are called **orthogonal polynomials**

Coefficients can be found for the special case of

- equal spaced levels (e.g., (0 15 30 45 60))
- equal sample sizes ($n_1 = n_2 = ... = n_{ab}$)

We can define the SS for a given contrast:

$$SS_L = \frac{\hat{L}^2}{\sum_{i=1}^{a}(c_i^2/bn_i)}$$

$$T = \frac{\hat{L}}{\sqrt{MSE\sum_{i=1}^{a}(c_i^2/bn_i)}} \sim t$$

Moreover,

$$t^2_{(1-\alpha/2;df)} = F_{(1-\alpha;1,df)}$$

So,

$$\frac{SS_L}{MSE} \sim F_{(1-\alpha;1,df_{MSE})}$$

all contrasts have d.f = 1

24.5.4.2 Unbalanced

We could have unequal numbers of replications for all treatment combinations:

- Observational studies
- Dropouts in designed studies
- Larger sample sizes for inexpensive treatments
- Sample sizes to match population makeup.

Assume that each factor combination has at least 1 observation (no empty cells)

Consider the same model as:

$$Y_{ijk} = \mu_{..} + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \epsilon_{ijk}$$

where sample sizes are: n_{ij}:

$$n_{i.} = \sum_j n_{ij}$$
$$n_{.j} = \sum_i n_{ij}$$
$$n_T = \sum_i \sum_j n_{ij}$$

Problem here is that

$$SSTO \neq SSA + SSB + SSAB + SSE$$

(the design is **non-orthogonal**)

- For $i = 1, ..., a-1$,

$$u_i = \begin{cases} +1 & \text{if the obs is from the i-th level of Factor 1} \\ -1 & \text{if the obs is from the a-th level of Factor 1} \\ 0 & \text{otherwise} \end{cases}$$

- For $j = 1, ..., b-1$

$$v_i = \begin{cases} +1 & \text{if the obs is from the j-th level of Factor 1} \\ -1 & \text{if the obs is from the b-th level of Factor 1} \\ 0 & \text{otherwise} \end{cases}$$

We can use these indicator variables as predictor variables and $\mu_{..}, \alpha_i, \beta_j, (\alpha\beta)_{ij}$ as unknown parameters.

$$Y = \mu_{..} + \sum_{i=1}^{a-1} \alpha_i u_i + \sum_{j=1}^{b-1} \beta_j v_j + \sum_{i=1}^{a-1} \sum_{j=1}^{b-1} (\alpha\beta)_{ij} u_i v_j + \epsilon$$

To test hypotheses, we use the extra sum of squares idea.

For interaction effects

$$H_0 : all (\alpha\beta)_{ij} = 0$$
$$H_a : \text{not all } (\alpha\beta)_{ij} = 0$$

Or to test

$$H_0 : \beta_1 = \beta_2 = \beta_3 = 0$$
$$H_a : \text{not all } \beta_j = 0$$

Analysis of Factor Means

(e.g., contrasts) is analogous to the balanced case, with modifications in the formulas for means and standard errors to account for unequal sample sizes.

Or , we can fit the cell means model and consider it from a regression perspective

If you have empty cells (i.e., some factor combinations have no observation), then the equivalent regression approach can't be used. But you can still do partial analyses

24.5.5 Two-Way Random Effects ANOVA

$$Y_{ijk} = \mu_{..} + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \epsilon_{ij}$$

where

- $\mu_{..}$: constant
- $\alpha_i \sim N(0, \sigma^2_\alpha), i = 1, .., a$ (independent)
- $\beta_j \sim N(0, \sigma^2_\beta), j = 1, .., b$ (independent)
- $(\alpha\beta)_{ij} \sim N(0, \sigma^2_{\alpha\beta}), i = 1, ..., a, j = 1, .., b$ (independent)
- $\epsilon_{ijk} \sim N(0, \sigma^2)$ (independent)

All $\alpha_i, \beta_j, (\alpha\beta)_{ij}$ are pairwise independent

Theoretical means, variances, and covariances are

$$E(Y_{ijk}) = \mu_{..}$$
$$var(Y_{ijk}) = \sigma^2_Y = \sigma^2_\alpha + \sigma^2_\beta + \sigma^2_{\alpha\beta} + \sigma^2$$

So

$Y_{ijk} \sim N(\mu_{..}, \sigma^2_\alpha + \sigma^2_\beta + \sigma^2_{\alpha\beta} + \sigma^2)$

$$cov(Y_{ijk}, Y_{ij'k'}) = \sigma^2_\alpha, j \neq j'$$
$$cov(Y_{ijk}, Y_{i'jk'}) = \sigma^2_\beta, i \neq i'$$
$$cov(Y_{ijk}, Y_{ijk'}) = \sigma^2_\alpha + \sigma^2_\beta + \sigma^2_{\alpha\beta}, k \neq k'$$
$$cov(Y_{ijk}, Y_{i'j'k'}) =, i \neq i', j \neq j'$$

24.5.6 Two-Way Mixed Effects ANOVA

24.5.6.1 Balanced

One fixed factor, while other is random treatment levels, we have a **mixed effects model** or a **mixed model**

Restricted mixed model for 2-way ANOVA:

$$Y_{ijk} = \mu_{..} + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \epsilon_{ijk}$$

where

- $\mu_{..}$: constant
- α_i: fixed effects with constraints subject to restriction $\sum \alpha_i = 0$
- $\beta_j \sim indepN(0, \sigma_\beta^2)$
- $(\alpha\beta)_{ij} \sim N(0, \frac{a-1}{a}\sigma_{\alpha\beta}^2)$ subject to restriction $\sum_i (\alpha\beta)_{ij} = 0$ for all j, the variance here is written as the proportion for convenience; it makes the expected mean squares simpler (other assumed $var((\alpha\beta)_{ij} = \sigma_{\alpha\beta}^2)$
- $cov((\alpha\beta)_{ij}, (\alpha\beta)_{i'j'}) = -\frac{1}{a}\sigma_{\alpha\beta}^2, i \neq i'$
- $\epsilon_{ijk} \sim indepN(0, \sigma^2)$
- $\beta_j, (\alpha\beta)_{ij}, \epsilon_{ijk}$ are pairwise independent

Two-way mixed models are written in an "unrestricted" form, with no restrictions on the interaction effects $(\alpha\beta)_{ij}$, they are pairwise independent.

Let $\beta^*, (\alpha\beta)_{ij}^*$ be the unrestricted random effects, and $(\bar{\alpha\beta})_{ij}^*$ the means averaged over the fixed factor for each level of random factor B.

$$\beta_j = \beta_j^* + (\bar{\alpha\beta})_{ij}^*$$
$$(\alpha\beta)_{ij} = (\alpha\beta)_{ij}^* - (\bar{\alpha\beta})_{ij}^*$$

Some consider the restricted model to be more general. but here we consider the restricted form.

$$E(Y_{ijk}) = \mu_{..} + \alpha_i$$
$$var(Y_{ijk}) = \sigma_\beta^2 + \frac{a-1}{a}\sigma_{\alpha\beta}^2 + \sigma^2$$

Responses from the same random factor (B) level are correlated

$$\begin{aligned} cov(Y_{ijk}, Y_{ijk'}) &= E(Y_{ijk}Y_{ijk'}) - E(Y_{ijk})E(Y_{ijk'}) \\ &= \sigma_\beta^2 + \frac{a-1}{a}\sigma_{\alpha\beta}^2, k \neq k' \end{aligned}$$

Similarly,

$$cov(Y_{ijk}, Y_{i'jk'}) = \sigma^2_\beta - \frac{1}{a}\sigma^2_{\alpha\ \beta}, i \neq i'$$
$$cov(Y_{ijk}, Y_{i'j'k'}) = 0, j \neq j'$$

Hence, you can see that the only way you don't have dependence in the Y is when they don't share the same random effect.

An advantage of the **restricted mixed model** is that 2 observations from the same random factor b level can be positively or negatively correlated. In the **unrestricted model**, they can only be positively correlated.

Mean Square	Fixed ANOVA (A, B Fixed)	Random ANOVA (A,B random)	Mixed ANVOA (A fixed, B random)
MSA	a - 1	$\sigma^2 + nb\frac{\sum \alpha_i^2}{a-1}$	$\sigma^2 + nb\sigma^2_\alpha + n\sigma^2_{\alpha\beta}$
MSB	b-1	$\sigma^2 + na\frac{\sum \beta_j^2}{b-1}$	$\sigma^2 + na\sigma^2_\beta + n\sigma^2_{\alpha\beta}$
MSAB	(a-1)(b-1)	$\sigma^2 + n\frac{\sum\sum(\alpha\beta)^2_{ij}}{(a-1)(b-1)}$	$\sigma^2 + n\sigma^2_{\alpha\beta}$
MSE	(n-1)ab	σ^2	σ^2

For fixed, random, and mixed models (balanced), the ANOVA table sums of squares calculations are identical. (also true for df and mean squares). The only difference is with the expected mean squares, thus the test statistics.

In Random ANOVA, we test

$$H_0 : \sigma^2 = 0$$
$$H_a : \sigma^2 > 0$$

by considering $F = \frac{MSA}{MSAB} \sim F_{a-1;(a-1)(b-1)}$

The same test statistic is used for mixed models, but in that case we are testing null hypothesis that all of the $\alpha_i = 0$

The test statistic different for the same null hypothesis under the fixed effects model.

Test for effects of	Fixed ANOVA (A&B fixed)	Random ANOVA (A&B random)	Mixed ANOVA (A fixed, B random)
Factor A	$\frac{MSA}{MSE}$	$\frac{MSA}{MSAB}$	$\frac{MSA}{MSAB}$
Factor B	$\frac{MSB}{MSE}$	$\frac{MSB}{MSAB}$	$\frac{MSB}{MSE}$
AB interactions	$\frac{MSAB}{MSE}$	$\frac{MSAB}{MSE}$	$\frac{MSAB}{MSE}$

Estimation Of Variance Components

In random and mixed effects models, we are interested in estimating the **variance components**
Variance component σ_β^2 in the mixed ANOVA.

$$E(\sigma_\beta^2) = \frac{E(MSB) - E(MSE)}{na} = \frac{\sigma^2 + na\sigma_\beta^2 - \sigma^2}{na} = \sigma_\beta^2$$

which can be estimated with

$$\hat{\sigma}_\beta^2 = \frac{MSB - MSE}{na}$$

Confidence intervals for variance components can be constructed (approximately) by using the **Satterthwaite** procedure or the MLS procedure (like the 1-way random effects)

Estimation of Fixed Effects in Mixed Models

$$\begin{aligned}
\hat{\alpha}_i &= \bar{Y}_{i..} - \bar{Y}_{...} \\
\hat{\mu}_{i.} &= \bar{Y}_{...} + (\bar{Y}_{i..} - \bar{Y}_{...}) = \bar{Y}_{i..} \\
\sigma^2(\hat{\alpha}_i) &= \frac{\sigma^2 + n\sigma_{\alpha\beta}^2}{bn} = \frac{E(MSAB)}{bn} \\
s^2(\hat{\alpha}_i) &= \frac{MSAB}{bn}
\end{aligned}$$

Contrasts on the **Fixed Effects**

$$\begin{aligned}
L &= \sum c_i \alpha_i \\
\sum c_i &= 0 \\
\hat{L} &= \sum c_i \hat{\alpha}_i \\
\sigma^2(\hat{L}) &= \sum c_i^2 \sigma^2(\hat{\alpha}_i) \\
s^2(\hat{L}) &= \frac{MSAB}{bn} \sum c_i^2
\end{aligned}$$

Confidence intervals and tests can be constructed as usual

24.5.6.2 Unbalanced

For a mixed model with a = 2, b = 4

$$\begin{aligned}
Y_{ijk} &= \mu_{..} + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \epsilon_{ijk} \\
var(\beta_j) &= \sigma_\beta^2 \\
var((\alpha\beta)_{ij}) &= \frac{2-1}{2}\sigma_{\alpha\beta}^2 = \frac{\sigma_{\alpha\beta}^2}{2} \\
var(\epsilon_{ijk}) &= \sigma^2 \\
E(Y_{ijk}) &= \mu_{..} + \alpha_i \\
var(Y_{ijk}) &= \sigma_\beta^2 + \frac{\sigma_{\alpha\beta}^2}{2} + \sigma^2 \\
cov(Y_{ijk}, Y_{ijk'}) &= \sigma^2 + \frac{\sigma_{\alpha\beta}^2}{2}, k \neq k' \\
cov(Y_{ijk}, Y_{i'jk'}) &= \sigma_\beta^2 - \frac{\sigma_{\alpha\beta}^2}{2}, i \neq i' \\
cov(Y_{ijk}, Y_{i'j'k'}) &= 0, j \neq j'
\end{aligned}$$

assume

$$\mathbf{Y} \sim N(\mathbf{X}\beta, M)$$

where M is block diagonal

density function

$$f(\mathbf{Y}) = \frac{1}{(2\pi)^{N/2}|M|^{1/2}} exp(-\frac{1}{2}(\mathbf{Y} - \mathbf{X}\ )'\mathbf{M^{-1}}(\mathbf{Y} - \mathbf{X}\ ))$$

if we knew the variance components, we could use GLS:

$$\hat{\beta}_{GLS} = (\mathbf{X'M^{-1}X})^{-\mathbf{1}}\mathbf{X'M^{-1}Y}$$

but we usually don't know the variance components $\sigma^2, \sigma_\beta^2, \sigma_{\alpha\beta}^2$ that make up M

Another way to get estimates is by **Maximum likelihood estimation**

we try to maximize its log

$$\ln L = -\frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln|M| - \frac{1}{2}(\mathbf{Y} - \mathbf{X}\ )'\ ^{-\mathbf{1}}(\mathbf{Y} - \mathbf{X}\ )$$

24.6 Nonparametric ANOVA

24.6.1 Kruskal-Wallis

Generalization of independent samples Wilcoxon Rank sum test for 2 independent samples (like F-test of one-way ANOVA is a generalization to several independent samples of the two sample t-test)

Consider the one-way case:

We have

- $a \geq 2$ treatments
- n_i is the sample size for the i-th treatment
- Y_{ij} is the j-th observation from the i-th treatment.
- we make **no** assumption of normality
- We only assume that observations on the i-th treatment are a random sample from the continuous CDF F_i, i = 1,..,n, and are mutually independent.

$$H_0 : F_1 = F_2 = ... = F_a$$
$$H_a : F_i < F_j \text{ for some } i \neq j$$

or if distribution is from the location-scale family, $H_0 : \theta_1 = \theta_2 = ... = \theta_a$)

Procedure

- Rank all $N = \sum_{i=1}^{a} n_i$ observations in ascending order. Let $r_{ij} = rank(Y_{ij})$, note $\sum_i \sum_j r_{ij} = 1 + 2.. + N = \frac{N(N+1)}{2}$

- Calculate the rank sums and averages:

$$r_{i.} = \sum_{j=1}^{n_i} r_{ij}$$

and

$$\bar{r}_{i.} = \frac{r_{i.}}{n_i}, i = 1, .., a$$

- Calculate the test statistic on the ranks:

$$\chi^2_{KW} = \frac{SSTR}{\frac{SSTO}{N-1}}$$

where $SSTR = \sum n_i(\bar{r}_{i.} - \bar{r}_{..})^2$ and $SSTO = \sum\sum(\bar{r}_{ij} - \bar{r}_{..})^2$
- For large n_i ($\geq$ 5 observations) the Kruskal-Wallis statistic is approximated by a χ^2_{a-1} distribution when all the treatment means are equal. Hence, reject H_0 if $\chi^2_{KW} > \chi^2_{(1-\alpha;a-1)}$.

- If sample sizes are small, one can exhaustively work out all possible distinct ways of assigning N ranks to the observations from a treatments and calculate the value of the KW statistic in each case ($\frac{N!}{n_1!..n_a!}$ possible combinations). Under H_0 all of these assignments are equally likely.

24.6.2 Friedman Test

When the responses $Y_{ij} = 1, .., n, j = 1, .., r$ in a randomized complete block design are not normally distributed (or do not have constant variance), a non-parametric test is more helpful.

A distribution-free rank-based test for comparing the treatments in this setting is the Friedman test. Let F_{ij} be the CDF of random Y_{ij}, corresponding to the observed value y_{ij}

Under the null hypothesis, F_{ij} are identical for all treatments j separately for each block i.

$$H_0 : F_{i1} = F_{i2} = ... = F_{ir} \text{ for all i}$$
$$H_a : F_{ij} < F_{ij'} \text{ for some } j \neq j' \text{ for all } i$$

For location parameter distributions, treatment effects can be tested:

$$H_0 : \tau_1 = \tau_2 = ... = \tau_r$$
$$H_a : \tau_j > \tau_{j'} \text{ for some } j \neq j'$$

Procedure

- Rank observations from the r treatments separately within each block (in ascending order; if ties, each tied observation is given the mean of ranks involved). Let the ranks be called r_{ij}

- Calculate the Friedman test statistic

$$\chi_F^2 = \frac{SSTR}{\frac{SSTR+SSE}{n(r-1)}}$$

where

$$SSTR = n\sum(\bar{r}_{.j} - \bar{r}_{..})^2$$
$$SSE = \sum\sum(r_{ij} - \bar{r}_{.j})^2$$
$$\bar{r}_{.j} = \frac{\sum_i r_{ij}}{n}$$
$$\bar{r}_{..} = \frac{r+1}{2}$$

If there is no ties, it can be rewritten as

$$\chi_F^2 = [\frac{12}{nr(n+1)} \sum_j r_{.j}^2] - 3n(r+1)$$

with large number of blocks, χ_F^2 is approximately χ_{r-1}^2 under H_0. Hence, we reject H_0 if $\chi_F^2 > \chi_{(1-\alpha;r-1)}^2$
The exact null distribution for χ_F^2 can be derived since there are r! possible ways of assigning ranks 1,2,...,r to the r observations within each block. There are n blocks and thus $(r!)^n$ possible assignments to the ranks, which are equally likely when H_0 is true.

24.7 Sample Size Planning for ANOVA

24.7.1 Balanced Designs

24.7.1.1 Single Factor Studies

24.7.1.1.1 Fixed cell means

$$P(F > f_{(1-\alpha;a-1,N-a)}|\phi) = 1 - \beta$$

where ϕ is the non-centrality **parameter** (measures how unequal the treatment means μ_i are)

$$\phi = \frac{1}{\sigma}\sqrt{\frac{n}{a}\sum_i(\mu_i - \mu_.)^2}, (n_i \equiv n)$$

and

$$\mu_. = \frac{\sum \mu_i}{a}$$

To decide on the power probabilities we use the non-central F distribution.

We could use the power table directly when effects are fixed and design is balanced by using **minimum range** of factor level means for your desired differences

$$\Delta = \max(\mu_i) - \min(\mu_i)$$

Hence, we need

- α level
- Δ

- σ
- β

Notes:

- When Δ/σ is small greatly affects sample size, but if Δ/σ is large.
- Reducing α or β increases the required sample sizes.
- Error in estimating σ can make a large difference.

24.7.1.2 Multi-factor Studies

The same noncentral F tables can be used here

For two-factor fixed effect model

Test for interactions:

$$\begin{aligned}
\phi &= \frac{1}{\sigma}\sqrt{\frac{n\sum\sum(\alpha\beta_{ij})^2}{(a-1)(b-1)+1}} = \frac{1}{\sigma}\sqrt{\frac{n\sum\sum(\mu_{ij}-\mu_{i.}-\mu_{.j}+\mu_{..})^2}{(a-1)(b-1)+1}} \\
\upsilon_1 &= (a-1)(b-1) \\
\upsilon_2 &= ab(n-1)
\end{aligned}$$

Test for Factor A main effects:

$$\begin{aligned}
\phi &= \frac{1}{\sigma}\sqrt{\frac{nb\sum\alpha_i^2}{a}} = \frac{1}{\sigma}\sqrt{\frac{nb\sum(\mu_{i.}-\mu_{..})^2}{a}} \\
\upsilon_1 &= a-1 \\
\upsilon_2 &= ab(n-1)
\end{aligned}$$

Test for Factor B main effects:

$$\begin{aligned}
\phi &= \frac{1}{\sigma}\sqrt{\frac{na\sum\beta_j^2}{b}} = \frac{1}{\sigma}\sqrt{\frac{na\sum(\mu_{.j}-\mu_{..})^2}{b}} \\
\upsilon_1 &= b-1 \\
\upsilon_2 &= ab(n-1)
\end{aligned}$$

Procedure:

1. Specify the minimum range of Factor A means
2. Obtain sample sizes with $r = a$. The resulting sample size is bn, from which n can be obtained.
3. Repeat the first 2 steps for Factor B minimum range.
4. Choose the greater number of sample size between A and B.

24.7.2 Randomized Block Experiments

Analogous to completely randomized designs . The power of the F-test for treatment effects for randomized block design uses the same non-centrality parameter as completely randomized design:

$$\phi = \frac{1}{\sigma}\sqrt{\frac{n}{r}\sum(\mu_i - \mu_.)^2}$$

However, the power level is different from the randomized block design because

- error variance σ^2 is different
- df(MSE) is different.

24.8 Randomized Block Designs

To improve the precision of treatment comparisons, we can reduce variability among the experimental units. We can group experimental units into **blocks** so that each block contains relatively homogeneous units.

- Within each block, random assignment treatments to units (separate random assignment for each block)
- The number of units per block is a multiple of the number of factor combinations.
- Commonly, use each treatment once in each block.

Benefits of **Blocking**

- Reduction in variability of estimators for treatment means
  - Improved power for t-tests and F-tests
  - Narrower confidence intervals
  - Smaller MSE
- Compare treatments under different conditions (related to different blocks).

Loss from **Blocking** (little to lose)

- If you don't do blocking well, you waste df on negligible block effects that could have been used to estimate σ^2
- Hence, the df for t-tests and denominator df for F-tests will be reduced without reducing MSE and small loss of power for both tests.

Consider

$$Y_{ij} = \mu_{..} + \rho_i + \tau_j + \epsilon_{ij}$$

where

- $i = 1, 2, \ldots, n$
- $j = 1, 2, \ldots, r$
- $\mu_{..}$: overall mean response, averaging across all blocks and treatments
- ρ_i: block effect, average difference in response for i-th block ($\sum \rho_i = 0$)
- τ_j treatment effect, average across blocks ($\sum \tau_j = 0$)
- $\epsilon_{ij} \sim iidN(0, \sigma^2)$: random experimental error.

Here, we assume that the block and treatment effects are additive. The difference in average response for any pair of treatments i the same **within** each block

$$(\mu_{..} + \rho_i + \tau_j) - (\mu_{..} + \rho_i + \tau'_j) = \tau_j - \tau'_j$$

for all $i = 1, .., n$ blocks

$$\begin{aligned}
\hat{\mu} &= \bar{Y}_{..} \\
\hat{\rho}_i &= \bar{Y}_{i.} - \bar{Y}_{..} \\
\hat{\tau}_j &= \bar{Y}_{.j} - \bar{Y}_{..}
\end{aligned}$$

Hence,

$$\begin{aligned}
\hat{Y}_{ij} &= \bar{Y}_{..} + (\bar{Y}_{i.} - \bar{Y}_{..}) + (\bar{Y}_{.j} - \bar{Y}_{..}) = \bar{Y}_{i.} + \bar{Y}_{.j} - \bar{Y}_{..} \\
e_{ij} &= Y_{ij} - \hat{Y}_{ij} = Y_{ij} - \bar{Y}_{i.} - \bar{Y}_{.j} + \bar{Y}_{..}
\end{aligned}$$

ANOVA table

Source of Variation	SS	df	Fixed Treatments E(MS)	Random Treatments E(MS)
Blocks	$r\sum_i(\bar{Y}_{i.} - \bar{Y}_{..})^2$	$n-1$	$\sigma^2 + r\frac{\sum \rho_i^2}{n-1}$	$\sigma^2 + r\frac{\sum \rho_i^2}{n-1}$
Treatments	$n\sum_j(\bar{Y}_{.j} - \bar{Y}_{..})^2$	$r-1$	$\sigma^2 + n\frac{\sum \tau_j^2}{r-1}$	$\sigma^2 + n\sigma_\tau^2$
Error	$\sum_i\sum_j(Y_{ij} - \bar{Y}_{i.} - \bar{Y}_{.j} + \bar{Y}_{..})^2$	$(n-1)(r-1)$	σ^2	σ^2
Total	$SSTO$	$nr-1$		

F-tests

$$H_0 : \tau_1 = \tau_2 = ... = \tau_r = 0 \qquad \text{Fixed Treatment Effects}$$
$$H_a : \text{not all } \tau_j = 0$$

$$H_0 : \sigma_\tau^2 = 0 \quad \text{Random Treatment Effects}$$
$$H_a : \sigma_\tau^2 \neq 0$$

In both cases $F = \frac{MSTR}{MSE}$, reject H_0 if $F > f_{(1-\alpha;r-1,(n-1)(r-1))}$

we don't use F-test to compare blocks, because

- We have a priori that blocs are different
- Randomization is done "within" block.

To estimate the efficiency that was gained by blocking (relative to completely randomized design).

$$\hat{\sigma}_{CR}^2 = \frac{(n-1)MSBL + n(r-1)MSE}{nr-1}$$
$$\hat{\sigma}_{RB}^2 = MSE$$
$$\frac{\hat{\sigma}_{CR}^2}{\hat{\sigma}_{RB}^2} = \text{above 1}$$

then a completely randomized experiment would

$$(\frac{\hat{\sigma}_{CR}^2}{\hat{\sigma}_{RB}^2} - 1)\%$$

more observations than the randomized block design to get the same MSE

If batches are randomly selected then they are random effects. That is , if the experiment was repeated, a new sample of i batches would be selected,d yielding new values for $\rho_1, \rho_2, ..., \rho_i$ then.

$$\rho_1, \rho_2, ..., \rho_j \sim N(0, \sigma_\rho^2)$$

Then,

$$Y_{ij} = \mu_{..} + \rho_i + \tau_j + \epsilon_{ij}$$

where

- $\mu_{..}$ fixed
- ρ_i: random iid $N(0, \sigma_p^2)$

- τ_j fixed (or random) $\sum \tau_j = 0$
- $\epsilon_{ij} \sim iidN(0, \sigma^2)$

Fixed Treatment

$$E(Y_{ij}) = \mu_{..} + \tau_j$$
$$var(Y_{ij}) = \sigma_\rho^2 + \sigma^2$$

$cov(Y_{ij}, Y_{ij'}) = \sigma^2, j \neq j'$ treatments within same block are correlated
$cov(Y_{ij}, Y_{i'j'}) = 0, i \neq i', j \neq j'$

Correlation between 2 observations in the same block

$$\frac{\sigma_\rho^2}{\sigma^2 + \sigma_\rho^2}$$

The expected MS for the additive fixed treatment effect, random block effect is

Source	SS	E(MS)
Blocks	SSBL	$\sigma^2 + r\sigma_\rho^2$
Treatment	SSTR	$\sigma^2 + n\frac{\sum \tau_j^2}{r-1}$
Error	SSE	σ^2

Interactions and Blocks
without replications within each block for each treatment, we can't consider interaction between block and treatment when the block effect is fixed. Hence, only in the random block effect, we have

$$Y_{ij} = \mu_{..} + \rho_i + \tau_j + (\rho\tau)_{ij} + \epsilon_{ij}$$

where

- $\mu_{..}$ constant
- $\rho_i \sim iddN(0, \sigma_\rho^2)$ random
- τ_j fixed ($\sum \tau_j = 0$)
- $(\rho\tau)_{ij} \sim N(0, \frac{r-1}{r}\sigma_{\rho\tau}^2)$ with $\sum_j (\rho\tau)_{ij} = 0$ for all i
- $cov((\rho\tau)_{ij}, (\rho\tau)_{ij'}) = -\frac{1}{r}\sigma_{\rho\tau}^2$ for $j \neq j'$
- $\epsilon_{ij} \sim iidN(0, \sigma^2)$ random

Note: a special case of mixed 2-factor model with 1 observation per "cell"

$$E(Y_{ij}) = \mu_{..} + \tau_j$$
$$var(Y_{ij}) = \sigma_\rho^2 + \frac{r-1}{r}\sigma_{\rho\tau}^2 + \sigma^2$$

$$cov(Y_{ij}, Y_{ij'}) = \sigma_\rho^2 - \frac{1}{r}\sigma_{\rho\tau}^2, j \neq j' \text{ obs from the same block are correlated}$$
$$cov(Y_{ij}, Y_{i'j'}) = 0, i \neq i', j \neq j' \text{ obs from different blocks are independent}$$

The sum of squares and degrees of freedom for interaction model are the same as those for the additive model. The difference exists only with the expected mean squares

Source	SS	df	E(MS)
Blocks	$SSBL$	$n-1$	$\sigma^2 + r\sigma_\rho^2$
Treatment	$SSTR$	$r-1$	$\sigma^2 + \sigma_{\rho\tau}^2 + n\frac{\sum \tau_j^2}{r-1}$
Error	SSE	$(n-1)(r-1)$	$\sigma^2 + \sigma_{\rho\tau}^2$

- No exact test is possible for block effects when interaction is present (Not important if blocks are used primarily to reduce experimental error variability)
- $E(MSE) = \sigma^2 + \sigma_{\rho\tau}^2$ the error term variance and interaction variance $\sigma_{\rho\tau}^2$. We can't estimate these components separately with this model. The two are **confounded**.
- If more than 1 observation per treatment block combination, one can consider interaction with fixed block effects, which is called **generalized randomized block designs** (multifactor analysis).

24.8.1 Tukey Test of Additivity

(Tukey's 1 df test for additivity)

formal test of interaction effects between blocks and treatments for a randomized block design. can also considered for testing additivity in 2-way analyses when there is only one observation per cell.

we consider a less restricted interaction term

$$(\rho\tau)_{ij} = D\rho_i\tau_j \text{(D: Constant)}$$

So,

$$Y_{ij} = \mu_{..} + \rho_i + \tau_j + D\rho_i\tau_j + \epsilon_{ij}$$

the least square estimate or MLE for D

$$\hat{D} = \frac{\sum_i \sum_j \rho_i \tau_j Y_{ij}}{\sum_i \rho_i^2 \sum_j \tau_j^2}$$

replacing the parameters by their estimates

$$\hat{D} = \frac{\sum_i \sum_j (\bar{Y}_{i.} - \bar{Y}_{..})(\bar{Y}_{.j} - \bar{Y}_{..})Y_{ij}}{\sum_i (\bar{Y}_{i.} - \bar{Y}_{..})^2 \sum_j (\bar{Y}_{.j} - \bar{Y}_{..})^2}$$

Thus, the interaction sum of squares

$$SSint = \sum_i \sum_j \hat{D}^2 (\bar{Y}_{i.} - \bar{Y}_{..})^2 (\bar{Y}_{.j} - \bar{Y}_{..})^2$$

The ANOVA decomposition

$$SSTO = SSBL + SSTR + SSint + SSRem$$

where $SSRem$: remainder sum of squares

$$SSRem = SSTO - SSBL - SSTR - SSint$$

if $D = 0$ (i.e., no interactions of the type $D\rho_i\tau_j$). $SSint$ and $SSRem$ are independent $\chi^2_{1,rn-r-n}$.

If $D = 0$,

$$F = \frac{SSint/1}{SSRem/(rn - r - n)} \sim f_{(1-\alpha;rn-r-n)}$$

if

$$H_0 : D = 0 \text{ no interaction present}$$
$$H_a : D \neq 0 \text{ interaction of form } D\rho_i\tau_j \text{ present}$$

we reject H_0 if $F > f_{(1-\alpha;1,nr-r-n)}$

24.9 Nested Designs

Let μ_{ij} be the mean response when factor A is at the i-th level and factor B is at the j-th level.
If the factors are crossed, the j-th level of B is the same for all levels of A.
If factor B is nested within A, the j-th level of B when A is at level 1 has nothing in common with the j-th level of B when A is at level 2.

Factors that can't be manipulated are designated as **classification factors**, as opposed to **experimental factors** (i.e., you assign to the experimental units).

24.9.1 Two-Factor Nested Designs

- Consider B is nested within A.
- both factors are fixed
- All treatment means are equally important.

Mean responses

$$\mu_{i.} = \sum_j \mu_{ij}/b$$

Main effect factor A

$$\alpha_i = \mu_{i.} - \mu_{..}$$

where $\mu_{..} = \frac{\mu_{ij}}{ab} = \frac{\sum_i \mu_{i.}}{a}$ and $\sum_i \alpha_i = 0$

Individual effects of B is denoted as $\beta_{j(i)}$ where $j(i)$ indicates the j-th level of factor B is nested within the it-h level of factor A

$$\begin{aligned} \beta_{j(i)} &= \mu_{ij} - \mu_{i.} \\ &= \mu_{ij} - \alpha_i - \mu_{..} \\ \sum_j \beta_{j(i)} &= 0, i = 1, ..., a \end{aligned}$$

$\beta_{j(i)}$ is the **specific effect** of the j-th level of factor B nested within the i-th level of factor A. Hence,

$$\mu_{ij} \equiv \mu_{..} + \alpha_i + \beta_{j(i)} \equiv \mu_{..} + (\mu_{i.} - \mu_{..}) + (\mu_{ij} - \mu_{i.})$$

Model

$$Y_{ijk} = \mu_{..} + \alpha_i + \beta_{j(i)} + \epsilon_{ijk}$$

where

- Y_{ijk} response for the k-th treatment when factor A is at the i-th level and factor B is at the j-th level ($i = 1, .., a; j = 1, .., b; k = 1, ..n$)
- $\mu_{..}$ constant
- α_i constants subject to restriction $\sum_i \alpha_i = 0$
- $\beta_{j(i)}$ constants subject to restriction $\sum_j \beta_{j(i)} = 0$ for all i
- $\epsilon_{ijk} \sim iidN(0, \sigma^2)$

$$E(Y_{ijk}) = \mu_{..} + \alpha_i + \beta_{j(i)}$$
$$var(Y_{ijk}) = \sigma^2$$

there is no interaction term in a nested model

ANOVA for Two-Factor Nested Designs

Least Squares and MLE estimates

Parameter	Estimator
$\mu_{..}$	$\bar{Y}_{...}$
α_i	$\bar{Y}_{i..} - \bar{Y}_{...}$
$\beta_{j(i)}$	$\bar{Y}_{ij.} - \bar{Y}_{i..}$
$\hat{Y}_{ijk}$	$\bar{Y}_{ij.}$

residual $e_{ijk} = Y_{ijk} - \bar{Y}_{ijk}$

$$SSTO = SSA + SSB(A) + SSE$$
$$\sum_i \sum_j \sum_k (Y_{ijk} - \bar{Y}_{...})^2 = bn \sum_i (\bar{Y}_{i..} - \bar{Y}_{...})^2 + n \sum_i \sum_j (\bar{Y}_{ij.} - \bar{Y}_{i..})^2 + \sum_i \sum_j \sum_k (Y_{ijk} - \bar{Y}_{ij.})^2$$

ANOVA Table

Source of Variation	SS	df	MS	E(MS)
Factor A	SSA	$a-1$	MSA	$\sigma^2 + bn\frac{\sum \alpha_i^2}{a-1}$
Factor B	$SSB(A)$	$a(b-1)$	$MSB(A)$	$\sigma^2 + n\frac{\|\|\|\|um \sum eta_{i)}^2}{a(b-1)}$
Error	SSE	$ab(n-1)$	MSE	σ^2
Total	$SSTO$	$abn-1$		

Tests For Factor Effects

$$H_0: \text{ All } \alpha_i = 0$$
$$H_a: \text{ not all } \alpha_i = 0$$

$F = \frac{MSA}{MSE} \sim f_{(1-\alpha;a-1,(n-1)ab)}$ reject if $F > f$

$$H_0: \text{ All } \beta_{j(i)} = 0$$
$$H_a: \text{ not all } \beta_{j(i)} = 0$$

$F = \frac{MSB(A)}{MSE} \sim f_{(1-\alpha;a(b-1),(n-1)ab)}$ reject $F > f$

Testing Factor Effect Contrasts

$L = \sum c_i \mu_i$ where $\sum c_i = 0$

$$\hat{L} = \sum c_i \bar{Y}_{i..}$$
$$\hat{L} \pm t_{(1-\alpha/2;df)} s(\hat{L})$$

where $s^2(\hat{L}) = \sum c_i^2 s^2(\bar{Y}_{i..})$, where $s^2(\bar{Y}_{i..}) = \frac{MSE}{bn}, df = ab(n-1)$

Testing Treatment Means

$L = \sum c_i \mu_{.j}$ estimated by $\hat{L} = \sum c_i \bar{Y}_{ij}$ with confidence limits:

$$\hat{L} \pm t_{(1-\alpha/2;(n-1)ab)} s(\hat{L})$$

where

$$s^2(\hat{L}) = \frac{MSE}{n} \sum c_i^2$$

Unbalanced Nested Two-Factor Designs

If there are different number of levels of factor B for different levels of factor A, then the design is called **unbalanced**

The model

$$
\begin{aligned}
Y_{ijk} &= \mu_{..} + \alpha_i + \beta_{j(i)} + \epsilon_{ijk} \\
\sum_{i=1}^{2} \alpha_i &= 0 \\
\sum_{j=1}^{3} \beta_{j(1)} &= 0 \\
\sum_{j=1}^{2} \beta_{j(2)} &= 0
\end{aligned}
$$

where

- $i = 1, 2; j = 1, .., b_i; k = 1, .., n_{ij}$
- $b_1 = 3, b_2 = 2, n_{11} = n_{13} = 2, n_{12} = 1, n_{21} = n_{22} = 2$
- $\alpha_1, \beta_{1(1)}, \beta_{2(1)}, \beta_{1(2)}$ are parameters.

And constraints: $\alpha_2 = -\alpha_1, \beta_{3(1)} = -\beta_{1(1)} - \beta_{2(1)}, \beta_{2(2)} = -\beta_{1(2)}$

4 indicator variables

$$
X_1 = \begin{cases} 1 & \text{if obs from school 1} \\ -1 & \text{if obs from school 2} \end{cases} \tag{24.5}
$$

$$
X_2 = \begin{cases} 1 & \text{if obs from instructor 1 in school 1} \\ -1 & \text{if obs from instructor 3 in school 1} \\ 0 & \text{otherwise} \end{cases} \tag{24.6}
$$

$$
X_3 = \begin{cases} 1 & \text{if obs from instructor 2 in school 1} \\ -1 & \text{if obs from instructor 3 in school 1} \\ 0 & \text{otherwise} \end{cases} \tag{24.7}
$$

$$
X_4 = \begin{cases} 1 & \text{if obs from instructor 1 in school 1} \\ -1 & \text{if obs from instructor 2 in school 1} \\ 0 & \text{otherwise} \end{cases} \tag{24.8}
$$

Regression Full Model

$$
Y_{ijk} = \mu_{..} + \alpha_1 X_{ijk1} + \beta_{1(1)} X_{ijk2} + \beta_{2(1)} X_{ijk3} + \beta_{1(2)} X_{ijk4} + \epsilon_{ijk}
$$

Random Factor Effects

If

$$\alpha_1 \sim iidN(0, \sigma_\alpha^2)$$
$$\beta_{j(i)} \sim iidN(0, \sigma_\beta^2)$$

Mean Square	Expected Mean Squares A fixed, B random	Expected Mean Squares A random, B random
MSA	$\sigma^2 + n\sigma_\beta^2 + bn\frac{\sum \alpha_i^2}{a-1}$	$\sigma^2 + bn\sigma_\alpha^2 + n\sigma_\beta^2$
MSB(A)	$\sigma^2 + n\sigma_\beta^2$	$\sigma^2 + n\sigma_\beta^2$
MSE	σ^2	σ^2

Test Statistics

Factor A	$\frac{MSA}{MSB(A)}$	$\frac{MSA}{MSB(A)}$
Factor B(A)	$\frac{MSB(A)}{MSE}$	$\frac{MSB(A)}{MSE}$

Another way to increase the precision of treatment comparisons by reducing variability is to use regression models to adjust for differences among experimental units (also known as **analysis of covariance**).

24.10 Single Factor Covariance Model

$$Y_{ij} = \mu_. + \tau_i + \gamma(X_{ij} - \bar{X}_{..}) + \epsilon_{ij}$$

for $i = 1, ..., r; j = 1, .., n_i$

where

- $\mu_.$ overall mean
- τ_i: fixed treatment effects ($\sum \tau_i = 0$)
- γ: fixed regression coefficient effect between X and Y
- X_{ij} covariate (not random)
- $\epsilon_{ij} \sim iidN(0, \sigma^2)$: random errors

If we just use γX_{ij} as the regression term (rather than $\gamma(X_{ij} - \bar{X}_{..})$), then $\mu_.$ is no longer the overall mean; thus we need to centered mean.

$$E(Y_{ij}) = \mu_. + \tau_i + \gamma(X_{ij} - \bar{X}_{..})$$
$$var(Y_{ij}) = \sigma^2$$

$Y_{ij} \sim N(\mu_{ij}, \sigma^2)$,

where

$$\mu_{ij} = \mu_{.} + \tau_i + \gamma(X_{ij} - \bar{X}_{..})$$
$$\sum \tau_i = 0$$

Thus, the mean response (μ_{ij}) is a regression line with intercept $\mu_{.} + \tau_i$ and slope γ for each treatment $$i.

Assumption:

- All treatment regression lines have the same slope
- when treatment interact with covariate X (non-parallel slopes), covariance analysis is **not** appropriate. in which case we should use separate regression lines.

More complicated regression features (e.g., quadratic, cubic) or additional covariates e.g.,

$$Y_{ij} = \mu_{.} + \tau_i + \gamma_1(X_{ij1} - \bar{X}_{..2}) + \gamma_2(X_{ij2} - \bar{X}_{..2}) + \epsilon_{ij}$$

Regression Formulation

We can use indicator variables for treatments

$$l_1 = \begin{cases} 1 & \text{if case is from treatment 1} \\ -1 & \text{if case is from treatment r} \\ 0 & \text{otherwise} \end{cases}$$

.

.

$$l_{r-1} = \begin{cases} 1 & \text{if case is from treatment r-1} \\ -1 & \text{if case is from treatment r} \\ 0 & \text{otherwise} \end{cases}$$

Let $x_{ij} = X_{ij} - \bar{X}_{..}$. the regression model is

$$Y_{ij} = \mu_{.} + \tau_1 l_{ij,1} + .. + \tau_{r-1} l_{ij,r-1} + \gamma x_{ij} + \epsilon_{ij}$$

where $I_{ij,1}$ is the indicator variable l_1 for the j-th case from treatment i. The treatment effect $\tau_1, ..\tau_{r-1}$ are just regression coefficients for the indicator variables.

We could use the same diagnostic tools for this case.

Inference

Treatment effects

$$H_0 : \tau_1 = \tau_2 = ... = 0$$
$$H_a : \text{not all } \tau_i = 0$$

$$\text{Full Model} : Y_{ij} = \mu_. + \tau_i + \gamma X_{ij} + \epsilon_{ij}$$
$$\text{Reduced Model} : Y_{ij} = \mu_. + \gamma X_{ij} + \epsilon_{ij}$$

$$F = \frac{SSE(R) - SSE(F)}{(N-2) - (N-(r+1))} / \frac{SSE(F)}{N-(r+1)} \sim F_{(r-1, N-(r+1))}$$

If we are interested in comparisons of treatment effects.
For example, r - 3. We estimate $\tau_1, \tau_2, \tau_3 = -\tau_1 - \tau_2$

Comparison	Estimate	Variance of Estimator
$\tau_1 - \tau_2$	$\hat{\tau}_1 - \hat{\tau}_2$	$var(\hat{\tau}_1) + var(\hat{\tau}_2) - 2cov(\hat{\tau}_1\hat{\tau}_2)$
$\tau_1 - \tau_3$	$2\hat{\tau}_1 + \hat{\tau}_2$	$4var(\hat{\tau}_1) + var(\hat{\tau}_2) - 4cov(\hat{\tau}_1\hat{\tau}_2)$
$\tau_2 - \tau_3$	$\hat{\tau}_1 + 2\hat{\tau}_2$	$var(\hat{\tau}_1) + 4var(\hat{\tau}_2) - 4cov(\hat{\tau}_1\hat{\tau}_2)$

Testing for Parallel Slopes

Example:

r = 3

$$Y_{ij} = \mu_. + \tau_1 I_{ij,1} + \tau_2 I_{ij,2} + \gamma X_{ij} + \beta_1 I_{ij,1} X_{ij} + \beta_2 I_{ij,2} X_{ij} + \epsilon_{ij}$$

where β_1, β_2: interaction coefficients.

$$H_0 : \beta_1 = \beta_2 = 0$$
$$H_a : \text{at least one} \beta \neq 0$$

If we can't reject H_0 using F-test then we have evidence that the slopes are parallel

Adjusted Means

The means in response after adjusting for the covariate effect

$$Y_{i.}(adj) = \bar{Y}_{i.} - \hat{\gamma}(\bar{X}_{i.} - \bar{X}_{..})$$

Chapter 25

Multivariate Methods

$y_1, ..., y_p$ are possibly correlated random variables with means $\mu_1, ..., \mu_p$

$$\mathbf{y} = \begin{pmatrix} y_1 \\ . \\ y_p \end{pmatrix}$$

$$E(\mathbf{y}) = \begin{pmatrix} \mu_1 \\ . \\ \mu_p \end{pmatrix}$$

Let $\sigma_{ij} = cov(y_i, y_j)$ for $i, j = 1, ..., p$

$$= (\sigma_{ij}) = \begin{pmatrix} \sigma_{11} & \sigma_{22} & ... & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & ... & \sigma_{2p} \\ . & . & . & . \\ \sigma_{p1} & \sigma_{p2} & ... & \sigma_{pp} \end{pmatrix}$$

where (symmetric) is the variance-covariance or dispersion matrix

Let $\mathbf{u}_{p\times 1}$ and $\mathbf{v}_{q\times 1}$ be random vectors with means μ_u and μ_v . Then

$$_{uv} = cov(\mathbf{u}, \mathbf{v}) = E[(\mathbf{u} - \mu_u)(\mathbf{v} - \mu_v)']$$

in which $_{uv} \neq {}_{vu}$ and $_{uv} = {}'_{vu}$

Properties of Covariance Matrices

1. Symmetric $' =$

2. Non-negative definite $\mathbf{a}'\ \mathbf{a} \geq 0$ for any $\mathbf{a} \in R^p$, which is equivalent to eigenvalues of , $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p \geq 0$
3. $|\ | = \lambda_1 \lambda_2 ... \lambda_p \geq 0$ (**generalized variance**) (the bigger this number is, the more variation there is
4. $trace(\ ) = tr(\ ) = \lambda_1 + ... + \lambda_p = \sigma_{11} + ... + \sigma_{pp}$ = sum of variance (**total variance**)

Note:

- is typically required to be positive definite, which means all eigenvalues are positive, and has an inverse $^{-1}$ such that $^{-1} = \mathbf{I}_{p \times p} = \ ^{-1}$

Correlation Matrices

$$\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}$$

$$\mathbf{R} = \begin{pmatrix} \rho_{11} & \rho_{12} & \cdots & \rho_{1p} \\ \rho_{21} & \rho_{22} & \cdots & \rho_{2p} \\ . & . & . & . \\ \rho_{p1} & \rho_{p2} & \cdots & \rho_{pp} \end{pmatrix}$$

where ρ_{ij} is the correlation, and $\rho_{ii} = 1$ for all i

Alternatively,

$$\mathbf{R} = [diag(\ )]^{-1/2}\ [diag(\ )]^{-1/2}$$

where $diag(\ )$ is the matrix which has the σ_{ii}'s on the diagonal and 0's elsewhere

and $\mathbf{A}^{1/2}$ (the square root of a symmetric matrix) is a symmetric matrix such as $\mathbf{A} = \mathbf{A}^{1/2}\mathbf{A}^{1/2}$

Equalities

Let

- $\mathbf{x}$ and $\mathbf{y}$ be random vectors with means μ_x and μ_y and variance -variance matrices $_x$ and $_y$.
- $\mathbf{A}$ and $\mathbf{B}$ be matrices of constants and $\mathbf{c}$ and $\mathbf{d}$ be vectors of constants

Then

- $E(\mathbf{Ay} + \mathbf{c}) = \mathbf{A}\mu_y + c$
- $var(\mathbf{Ay} + \mathbf{c}) = \mathbf{A}var(\mathbf{y})\mathbf{A}' = \mathbf{A}\ _{\mathbf{y}}\mathbf{A}'$
- $cov(\mathbf{Ay} + \mathbf{c}, \mathbf{By} + \mathbf{d}) = \mathbf{A}\ _{\mathbf{y}}\mathbf{B}'$
- $E(\mathbf{Ay} + \mathbf{Bx} + \mathbf{c}) = \mathbf{A}\ _{\mathbf{y}} + \mathbf{B}\ _{\mathbf{x}} + \mathbf{c}$

- $var(\mathbf{Ay} + \mathbf{Bx} + \mathbf{c}) = \mathbf{A}\ _{\mathbf{y}}\mathbf{A}' + \mathbf{B}\ _{\mathbf{x}}\mathbf{B}' + \mathbf{A}\ _{\mathbf{yx}}\mathbf{B}' + \mathbf{B}\ '_{\mathbf{yx}}\mathbf{A}'$

Multivariate Normal Distribution

Let $\mathbf{y}$ be a multivariate normal (MVN) random variable with mean μ and variance . Then the density of $\mathbf{y}$ is

$$f(\mathbf{y}) = \frac{1}{(2\pi)^{p/2}|\ |^{1/2}} \exp(-\frac{1}{2}(\mathbf{y} - \ )'\ ^{-\mathbf{1}}(\mathbf{y} - \ ))$$

$\mathbf{y} \sim N_p(\mu,\ )$

25.0.1 Properties of MVN

- Let $\mathbf{A}_{r\times p}$ be a fixed matrix. Then $\mathbf{Ay} \sim N_r(\mathbf{A}\ , \mathbf{A}\ \mathbf{A}')$. $r \leq p$ and all rows of $\mathbf{A}$ must be linearly independent to guarantee that $\mathbf{A}\ \mathbf{A}'$ is non-singular.
- Let $\mathbf{G}$ be a matrix such that $^{-1} = \mathbf{GG}'$. Then $\mathbf{G}'\mathbf{y} \sim N_p(\mathbf{G}'\ , \mathbf{I})$ and $\mathbf{G}'(\mathbf{y} - \ ) \sim N_p(0, \mathbf{I})$
- Any fixed linear combination of $y_1, ..., y_p$ (say $\mathbf{c}'\mathbf{y}$) follows $\mathbf{c}'\mathbf{y} \sim N_1(\mathbf{c}'\ , \mathbf{c}'\ \mathbf{c})$
- Define a partition, $[\mathbf{y}_1', \mathbf{y}_2']'$ where
  - $\mathbf{y}_1$ is $p_1 \times 1$
  - $\mathbf{y}_2$ is $p_2 \times 1$,
  - $p_1 + p_2 = p$
  - $p_1, p_2 \geq 1$ Then

$$\begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix} \sim N\left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} {}_{11} & {}_{12} \\ {}_{21} & {}_{22} \end{pmatrix}\right)$$

- The marginal distributions of $\mathbf{y}_1$ and $\mathbf{y}_2$ are $\mathbf{y}_1 \sim N_{p1}(\ _{\mathbf{1}},\ _{\mathbf{11}})$ and $\mathbf{y}_2 \sim N_{p2}(\ _{\mathbf{2}},\ _{\mathbf{22}})$
- Individual components $y_1, ..., y_p$ are all normally distributed $y_i \sim N_1(\mu_i, \sigma_{ii})$
- The conditional distribution of $\mathbf{y}_1$ and $\mathbf{y}_2$ is normal
  - $\mathbf{y}_1|\mathbf{y}_2 \sim N_{p1}(\ _{\mathbf{1}} +\ _{\mathbf{12}}\ ^{\mathbf{-1}}_{\mathbf{22}}(\mathbf{y}_{\mathbf{2}} -\ _{\mathbf{2}}),\ _{\mathbf{11}} -\ _{\mathbf{12}}\ ^{\mathbf{-1}}_{\mathbf{22}}\ _{\mathbf{21}})$
    - In this formula, we see if we know (have info about) $\mathbf{y}_2$, we can re-weight $\mathbf{y}_1$'s mean, and the variance is reduced because we know more about $\mathbf{y}_1$ because we know $\mathbf{y}_2$
  - which is analogous to $\mathbf{y}_2|\mathbf{y}_1$. And $\mathbf{y}_1$ and $\mathbf{y}_2$ are independently distrusted only if $_{12} = 0$

- If $\mathbf{y} \sim N(\ ,\ )$ and is positive definite, then $(\mathbf{y} - \ )' \ ^{-\mathbf{1}}(\mathbf{y} - \ ) \sim \chi^2_{(p)}$
- If $\mathbf{y}_i$ are independent $N_p(\ _i,\ _i)$ random variables, then for fixed matrices $\mathbf{A}_{i(m\times p)}$, $\sum_{i=1}^k \mathbf{A}_i\mathbf{y}_i \sim N_m(\sum_{i=1}^k \mathbf{A}_i\ _i, \sum_{i=1}^k \mathbf{A}_i\ _i\mathbf{A}_i)$

Multiple Regression

$$\begin{pmatrix} Y \\ \mathbf{x} \end{pmatrix} \sim N_{p+1}\left(\begin{bmatrix} \mu_y \\ \ _x \end{bmatrix}, \begin{bmatrix} \sigma^2_Y & \ _{yx} \\ \ _{yx} & \ _{xx} \end{bmatrix}\right)$$

The conditional distribution of Y given x follows a univariate normal distribution with

$$\begin{aligned} E(Y|\mathbf{x}) &= \mu_y + \ _{yx}\Sigma^{-1}_{xx}(\mathbf{x} - \mu_x) \\ &= \mu_y - \Sigma_{yx}\Sigma^{-1}_{xx}\mu_x + \Sigma_{yx}\Sigma^{-1}_{xx}\mathbf{x} \\ &= \beta_0 + \ '\mathbf{x} \end{aligned}$$

where $\beta = (\beta_1, ..., \beta_p)' = \ ^{-1}_{xx}\ ^{\prime}_{yx}$ (e.g., analogous to $(\mathbf{x}'\mathbf{x})^{-\mathbf{1}}\mathbf{x}'\mathbf{y}$ but not the same if we consider Y_i and $\mathbf{x}_i$, $i = 1, .., n$ and use the empirical covariance formula: $var(Y|\mathbf{x}) = \sigma^2_Y - \ _{\mathbf{yx}}\ ^{-\mathbf{1}}_{\mathbf{xx}}\ ^{\prime}_{\mathbf{yx}})$

Samples from Multivariate Normal Populations

A random sample of size n, $\mathbf{y}_1, .., \mathbf{y}_n$ from $N_p(\ ,\ )$. Then

- Since $\mathbf{y}_1, ..., \mathbf{y}_n$ are iid, their sample mean, $\bar{\mathbf{y}} = \sum_{i=1}^n \mathbf{y}_i/n \sim N_p(\ ,\ /n)$. that is, $\bar{\mathbf{y}}$ is an unbiased estimator of
- The $p \times p$ sample variance-covariance matrix, $\mathbf{S}$ is $\mathbf{S} = \frac{1}{n-1}\sum_{i=1}^n(\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})' = \frac{1}{n-1}(\sum_{i=1}^n \mathbf{y}_i\mathbf{y}_i' - n\bar{\mathbf{y}}\bar{\mathbf{y}}')$
  - where $\mathbf{S}$ is symmetric, unbiased estimator of and has $p(p+1)/2$ random variables.
- $(n-1)\mathbf{S} \sim W_p(n-1,\ )$ is a Wishart distribution with n-1 degrees of freedom and expectation $(n-1)$. The Wishart distribution is a multivariate extension of the Chi-squared distribution.
- $\bar{\mathbf{y}}$ and $\mathbf{S}$ are independent
- $\bar{\mathbf{y}}$ and $\mathbf{S}$ are sufficient statistics. (All of the info in the data about and is contained in $\bar{\mathbf{y}}$ and $\mathbf{S}$, regardless of sample size).

Large Sample Properties

$\mathbf{y}_1, ..., \mathbf{y}_n$ are a random sample from some population with mean and variance-covariance matrix

- $\bar{\mathbf{y}}$ is a consistent estimator for μ
- $\mathbf{S}$ is a consistent estimator for

- **Multivariate Central Limit Theorem**: Similar to the univariate case, $\sqrt{n}(\bar{\mathbf{y}} - \mu) \dot{\sim} N_p(\mathbf{0}, \ )$ where n is large relative to p ($n \geq 25p$), which is equivalent to $\bar{\mathbf{y}} \dot{\sim} N_p(\mu, \ /n)$
- **Wald's Theorem**: $n(\bar{\mathbf{y}} - \mu)' \mathbf{S}^{-1} (\bar{\mathbf{y}} - \mu)$ when n is large relative to p.

Maximum Likelihood Estimation for MVN

Suppose iid $\mathbf{y}_1, ...\mathbf{y}_n \sim N_p(\mu, \ )$, the likelihood function for the data is

$$L(\mu, \ ) = \prod_{j=1}^{n} (\frac{1}{(2\pi)^{p/2} | \ |^{1/2}} \exp(-\frac{1}{2}(\mathbf{y}_j - \mu)' \ ^{-1})(\mathbf{y}_j - \mu))$$
$$= \frac{1}{(2\pi)^{np/2} | \ |^{n/2}} \exp(-\frac{1}{2} \sum_{j=1}^{n} (\mathbf{y}_j - \mu)' \ ^{-1})(\mathbf{y}_j - \mu)$$

Then, the MLEs are

$$\hat{\mu} = \bar{\mathbf{y}}$$

$$\hat{} = \frac{n-1}{n} \mathbf{S}$$

using derivatives of the log of the likelihood function with respect to μ and

Properties of MLEs

- Invariance: If $\hat{\theta}$ is the MLE of θ, then the MLE of $h(\theta)$ is $h(\hat{\theta})$ for any function h(.)
- Consistency: MLEs are consistent estimators, but they are usually biased
- Efficiency: MLEs are efficient estimators (no other estimator has a smaller variance for large samples)
- Asymptotic normality: Suppose that $\hat{\theta}_n$ is the MLE for θ based upon n independent observations. Then $\hat{\theta}_n \dot{\sim} N(\theta, \mathbf{H}^{-1})$
  - $\mathbf{H}$ is the Fisher Information Matrix, which contains the expected values of the second partial derivatives fo the log-likelihood function. the (i,j)th element of $\mathbf{H}$ is $-E(\frac{\partial^2 l(\ )}{\partial \theta_i \partial \theta_j})$
  - we can estimate $\mathbf{H}$ by finding the form determined above, and evaluate it at $\theta = \hat{\theta}_n$
- Likelihood ratio testing: for some null hypothesis, H_0 we can form a likelihood ratio test
  - The statistic is: $\Lambda = \frac{\max_{H_0} l(\ , |\mathbf{Y})}{\max l(\mu, |\mathbf{Y})}$

  - For large n, $-2\log\Lambda \sim \chi^2_{(v)}$ where v is the number of parameters in the unrestricted space minus the number of parameters under H_0

Test of Multivariate Normality

- Check univariate normality for each trait (X) separately
  - Can check

$$NormalityAssessment$$

  - The good thing is that if any of the univariate trait is not normal, then the joint distribution is not normal (see again [m]). If a joint multivariate distribution is normal, then the marginal distribution has to be normal.
  - However, marginal normality of all traits does not imply joint MVN
  - Easily rule out multivariate normality, but not easy to prove it
- Mardia's tests for multivariate normality
  - Multivariate skewness is

$$\beta_{1,p} = E[(\mathbf{y} - \ )'\ ^{-1}(\mathbf{x} - \ )]^3$$

  - where $\mathbf{x}$ and $\mathbf{y}$ are independent, but have the same distribution (note: β here is not regression coefficient)
  - Multivariate kurtosis is defined as
  -

$$\beta_{2,p} - E[(\mathbf{y} - \ )'\ ^{-1}(\mathbf{x} - \ )]^2$$

  - For the MVN distribution, we have $\beta_{1,p} = 0$ and $\beta_{2,p} = p(p+2)$
  - For a sample of size n, we can estimate

$$\hat{\beta}_{1,p} = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij}^2$$

$$\hat{\beta}_{2,p} = \frac{1}{n}\sum_{i=1}^{n} g_{ii}^2$$

    * where $g_{ij} = (\mathbf{y}_i - \bar{\mathbf{y}})'\mathbf{S}^{-1}(\mathbf{y}_j - \bar{\mathbf{y}})$. Note: $g_{ii} = d_i^2$ where d_i^2 is the Mahalanobis distance
  - (Mardia, 1970) shows for large n

$$\kappa_1 = \frac{n\hat{\beta}_{1,p}}{6} \dot{\sim} \chi^2_{p(p+1)(p+2)/6}$$

$$\kappa_2 = \frac{\hat{\beta}_{2,p} - p(p+2)}{\sqrt{8p(p+2)/n}} \sim N(0,1)$$

  * Hence, we can use κ_1 and κ_2 to test the null hypothesis of MVN.
  * When the data are non-normal, normal theory tests on the mean are sensitive to $\beta_{1,p}$, while tests on the covariance are sensitive to $\beta_{2,p}$

- Alternatively, Doornik-Hansen test for multivariate normality (Doornik and Hansen, 2008)
- Chi-square Q-Q plot
  - Let $\mathbf{y}_i, i = 1, ..., n$ be a random sample sample from $N_p(\ , \ )$
  - Then $\mathbf{z}_i = \ ^{-1/2}(\mathbf{y}_i - \ ), i = 1, ..., n$ are iid $N_p(\mathbf{0}, \mathbf{I})$. Thus, $d_i^2 = \mathbf{z}_i'\mathbf{z}_i \sim \chi_p^2, i = 1, ..., n$
  - plot the ordered d_i^2 values against the qualities of the χ_p^2 distribution. When normality holds, the plot should approximately resemble a straight lien passing through the origin at a 45 degree
  - it requires large sample size (i.e., sensitive to sample size). Even if we generate data from a MVN, the tail of the Chi-square Q-Q plot can still be out of line.
- If the data are not normal, we can
  - ignore it
  - use nonparametric methods
  - use models based upon an approximate distribution (e.g., GLMM)
  - try performing a transformation

```
library(heplots)
library(ICSNP)
library(MVN)
library(tidyverse)

trees = read.table("images/trees.dat")
names(trees) <- c("Nitrogen","Phosphorous","Potassium","Ash","Height")
str(trees)
#> 'data.frame':    26 obs. of  5 variables:
#>  $ Nitrogen   : num  2.2 2.1 1.52 2.88 2.18 1.87 1.52 2.37 2.06 1.84 ...
#>  $ Phosphorous: num  0.417 0.354 0.208 0.335 0.314 0.271 0.164 0.302 0.373 0.265 ...
#>  $ Potassium  : num  1.35 0.9 0.71 0.9 1.26 1.15 0.83 0.89 0.79 0.72 ...
#>  $ Ash        : num  1.79 1.08 0.47 1.48 1.09 0.99 0.85 0.94 0.8 0.77 ...
#>  $ Height     : int  351 249 171 373 321 191 225 291 284 213 ...
```

```r
summary(trees)
#>     Nitrogen      Phosphorous       Potassium           Ash
#>  Min.   :1.130   Min.   :0.1570   Min.   :0.3800   Min.   :0.4500
#>  1st Qu.:1.532   1st Qu.:0.1963   1st Qu.:0.6050   1st Qu.:0.6375
#>  Median :1.855   Median :0.2250   Median :0.7150   Median :0.9300
#>  Mean   :1.896   Mean   :0.2506   Mean   :0.7619   Mean   :0.8873
#>  3rd Qu.:2.160   3rd Qu.:0.2975   3rd Qu.:0.8975   3rd Qu.:0.9825
#>  Max.   :2.880   Max.   :0.4170   Max.   :1.3500   Max.   :1.7900
#>      Height
#>  Min.   : 65.0
#>  1st Qu.:122.5
#>  Median :181.0
#>  Mean   :196.6
#>  3rd Qu.:276.0
#>  Max.   :373.0
cor(trees, method = "pearson") # correlation matrix
#>              Nitrogen Phosphorous Potassium       Ash    Height
#> Nitrogen    1.0000000   0.6023902 0.5462456 0.6509771 0.8181641
#> Phosphorous 0.6023902   1.0000000 0.7037469 0.6707871 0.7739656
#> Potassium   0.5462456   0.7037469 1.0000000 0.6710548 0.7915683
#> Ash         0.6509771   0.6707871 0.6710548 1.0000000 0.7676771
#> Height      0.8181641   0.7739656 0.7915683 0.7676771 1.0000000

# qq-plot
gg <- trees %>%
    pivot_longer(everything(), names_to = "Var", values_to = "Value") %>%
    ggplot(aes(sample = Value)) +
    geom_qq() +
    geom_qq_line() +
    facet_wrap("Var", scales = "free")
gg
```

```
# Univariate normality
sw_tests <- apply(trees, MARGIN = 2, FUN = shapiro.test)
sw_tests
#> $Nitrogen
#>
#>  Shapiro-Wilk normality test
#>
#> data:  newX[, i]
#> W = 0.96829, p-value = 0.5794
#>
#>
#> $Phosphorous
#>
#>  Shapiro-Wilk normality test
#>
#> data:  newX[, i]
#> W = 0.93644, p-value = 0.1104
#>
#>
#> $Potassium
#>
#>  Shapiro-Wilk normality test
#>
#> data:  newX[, i]
#> W = 0.95709, p-value = 0.3375
#>
```

```
#>
#> $Ash
#>
#>  Shapiro-Wilk normality test
#>
#> data:  newX[, i]
#> W = 0.92071, p-value = 0.04671
#>
#>
#> $Height
#>
#>  Shapiro-Wilk normality test
#>
#> data:  newX[, i]
#> W = 0.94107, p-value = 0.1424
# Kolmogorov-Smirnov test
ks_tests <- map(trees, ~ ks.test(scale(.x),"pnorm"))
ks_tests
#> $Nitrogen
#>
#>  Asymptotic one-sample Kolmogorov-Smirnov test
#>
#> data:  scale(.x)
#> D = 0.12182, p-value = 0.8351
#> alternative hypothesis: two-sided
#>
#>
#> $Phosphorous
#>
#>  Asymptotic one-sample Kolmogorov-Smirnov test
#>
#> data:  scale(.x)
#> D = 0.17627, p-value = 0.3944
#> alternative hypothesis: two-sided
#>
#>
#> $Potassium
#>
#>  Asymptotic one-sample Kolmogorov-Smirnov test
#>
#> data:  scale(.x)
#> D = 0.10542, p-value = 0.9348
#> alternative hypothesis: two-sided
#>
#>
```

```
#> $Ash
#>
#>  Asymptotic one-sample Kolmogorov-Smirnov test
#>
#> data:  scale(.x)
#> D = 0.14503, p-value = 0.6449
#> alternative hypothesis: two-sided
#>
#>
#> $Height
#>
#>  Asymptotic one-sample Kolmogorov-Smirnov test
#>
#> data:  scale(.x)
#> D = 0.1107, p-value = 0.9076
#> alternative hypothesis: two-sided

# Mardia's test, need large sample size for power
mardia_test <-
    mvn(
        trees,
        mvnTest = "mardia",
        covariance = FALSE,
        multivariatePlot = "qq"
    )

mardia_test$multivariateNormality
#>              Test         Statistic            p value Result
#> 1 Mardia Skewness  29.7248528871795   0.720544267457 78    YES
#> 2 Mardia Kurtosis -1.67743173185383 0.0934580886477281    YES
#> 3             MVN              <NA>               <NA>    YES

# Doornik-Hansen's test
dh_test <-
    mvn(
        trees,
        mvnTest = "dh",
        covariance = FALSE,
        multivariatePlot = "qq"
    )
```

```
dh_test$multivariateNormality
#>              Test        E df      p value MVN
#> 1 Doornik-Hansen 161.9446 10 1.285352e-29  NO

# Henze-Zirkler's test
hz_test <-
    mvn(
        trees,
        mvnTest = "hz",
        covariance = FALSE,
        multivariatePlot = "qq"
    )
hz_test$multivariateNormality
#>             Test        HZ   p value MVN
#> 1 Henze-Zirkler 0.7591525 0.6398905 YES
# The last column indicates whether dataset follows a multivariate normality or not (i

# Royston's test
# can only apply for 3 < obs < 5000 (because of Shapiro-Wilk's test)
royston_test <-
    mvn(
        trees,
        mvnTest = "royston",
        covariance = FALSE,
        multivariatePlot = "qq"
    )
royston_test$multivariateNormality
```

```
#>      Test        H    p value MVN
#> 1 Royston 9.064631 0.08199215 YES


# E-statistic
estat_test <-
    mvn(
        trees,
        mvnTest = "energy",
        covariance = FALSE,
        multivariatePlot = "qq"
    )
estat_test$multivariateNormality
#>          Test Statistic p value MVN
#> 1 E-statistic  1.091101   0.519 YES
```

25.0.2 Mean Vector Inference

In the univariate normal distribution, we test $H_0 : \mu = \mu_0$ by using

$$T = \frac{\bar{y} - \mu_0}{s/\sqrt{n}} \sim t_{n-1}$$

under the null hypothesis. And reject the null if $|T|$ is large relative to $t_{(1-\alpha/2, n-1)}$ because it means that seeing a value as large as what we observed is rare if the null is true

Equivalently,

$$T^2 = \frac{(\bar{y} - \mu_0)^2}{s^2/n} = n(\bar{y} - \mu_0)(s^2)^{-1}(\bar{y} - \mu_0) \sim f_{(1,n-1)}$$

25.0.2.1 Natural Multivariate Generalization

$$\begin{aligned} H_0 : & \ = \ _0 \\ H_a : & \ \neq \ _0 \end{aligned}$$

Define **Hotelling's** T^2 by

$$T^2 = n(\bar{\mathbf{y}} - \ _0)'\mathbf{S}^{-1}(\bar{\mathbf{y}} - \ _0)$$

which can be viewed as a generalized distance between $\bar{\mathbf{y}}$ and $\ _0$

Under the assumption of normality,

$$F = \frac{n-p}{(n-1)p} T^2 \sim f_{(p,n-p)}$$

and reject the null hypothesis when $F > f_{(1-\alpha,p,n-p)}$

- The T^2 test is invariant to changes in measurement units.
  - If $\mathbf{z} = \mathbf{C}\mathbf{y} + \mathbf{d}$ where $\mathbf{C}$ and $\mathbf{d}$ do not depend on $\mathbf{y}$, then $T^2(\mathbf{z}) - T^2(\mathbf{y})$
- The T^2 test can be derived as a **likelihood ratio** test of $H_0 : \mu = \mu_0$

25.0.2.2 Confidence Intervals

25.0.2.2.1 Confidence Region An "exact" $100(1-\alpha)\%$ confidence region for is the set of all vectors, $\mathbf{v}$, which are "close enough" to the observed mean vector, $\bar{\mathbf{y}}$ to satisfy

$$n(\bar{\mathbf{y}} - {}_0)'\mathbf{S}^{-1}(\bar{\mathbf{y}} - {}_0) \leq \frac{(n-1)p}{n-p} f_{(1-\alpha,p,n-p)}$$

- $\mathbf{v}$ are just the mean vectors that are not rejected by the T^2 test when $\bar{\mathbf{y}}$ is observed.

In case that you have 2 parameters, the confidence region is a "hyper-ellipsoid".

In this region, it consists of all ${}_0$ vectors for which the T^2 test would not reject H_0 at significance level α

Even though the confidence region better assesses the joint knowledge concerning plausible values of , people typically include confidence statement about the individual component means. We'd like all of the separate confidence statements to hold **simultaneously** with a specified high probability. Simultaneous confidence intervals: intervals **against** any statement being incorrect

25.0.2.2.1.1 Simultaneous Confidence Statements

- Intervals based on a rectangular confidence region by projecting the previous region onto the coordinate axes:

$$\bar{y}_i \pm \sqrt{\frac{(n-1)p}{n-p} f_{(1-\alpha,p,n-p)} \frac{s_{ii}}{n}}$$

for all $i = 1, .., p$

which implied confidence region is conservative; it has at least $100(1-\alpha)\%$

Generally, simultaneous $100(1-\alpha)\%$ confidence intervals for all linear combinations , $\mathbf{a}$ of the elements of the mean vector are given by

$$\mathbf{a}'\bar{\mathbf{y}} \pm \sqrt{\frac{(n-1)p}{n-p} f_{(1-\alpha,p,n-p)} \frac{\mathbf{a}'\mathbf{S}\mathbf{a}}{n}}$$

- works for any arbitrary linear combination $\mathbf{a}' = a_1\mu_1 + ... + a_p\mu_p$, which is a projection onto the axis in the direction of $\mathbf{a}$
- These intervals have the property that the probability that at least one such interval does not contain the appropriate $\mathbf{a}'$ is no more than α
- These types of intervals can be used for "data snooping" (like

  $$Scheffe$$

)

25.0.2.2.1.2 One μ at a time

- One at a time confidence intervals:

  $$\bar{y}_i \pm t_{(1-\alpha/2,n-1} \sqrt{\frac{s_{ii}}{n}}$$

- Each of these intervals has a probability of $1-\alpha$ of covering the appropriate μ_i
- But they ignore the covariance structure of the p variables
- If we only care about k simultaneous intervals, we can use "one at a time" method with the

  $$Bonferroni$$

  correction.
- This method gets more conservative as the number of intervals k increases.

25.0.3 General Hypothesis Testing

25.0.3.1 One-sample Tests

$$H_0 : \mathbf{C} = \mathbf{0}$$

where

- $\mathbf{C}$ is a $c \times p$ matrix of rank c where $c \leq p$

We can test this hypothesis using the following statistic

$$F = \frac{n-c}{(n-1)c} T^2$$

where $T^2 = n(\mathbf{C}\bar{\mathbf{y}})'(\mathbf{CSC}')^{-1}(\mathbf{C}\bar{\mathbf{y}})$

Example:

$$H_0 : \mu_1 = \mu_2 = ... = \mu_p$$

Equivalently,

$$\begin{aligned} \mu_1 - \mu_2 &= 0 \\ &\vdots \\ \mu_{p-1} - \mu_p &= 0 \end{aligned}$$

a total of $p-1$ tests. Hence, we have $\mathbf{C}$ as the $p-1 \times p$ matrix

$$\mathbf{C} = \begin{pmatrix} 1 & -1 & 0 & ... & 0 \\ 0 & 1 & -1 & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & 1 & -1 \end{pmatrix}$$

number of rows $= c = p-1$

Equivalently, we can also compare all of the other means to the first mean. Then, we test $\mu_1 - \mu_2 = 0, \mu_1 - \mu_3 = 0, ..., \mu_1 - \mu_p = 0$, the $(p-1) \times p$ matrix $\mathbf{C}$ is

$$\mathbf{C} = \begin{pmatrix} -1 & 1 & 0 & ... & 0 \\ -1 & 0 & 1 & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & ... & 0 & 1 \end{pmatrix}$$

The value of T^2 is invariant to these equivalent choices of $\mathbf{C}$

This is often used for **repeated measures designs**, where each subject receives each treatment once over successive periods of time (all treatments are administered to each unit).

Example:

Let y_{ij} be the response from subject i at time j for $i = 1, .., n, j = 1, ..., T$. In this case, $\mathbf{y}_i = (y_{i1}, ..., y_{iT})', i = 1, ..., n$ are a random sample from $N_T(\ ,\ )$

Let $n = 8$ subjects, $T = 6$. We are interested in $\mu_1, .., \mu_6$

$$H_0 : \mu_1 = \mu_2 = ... = \mu_6$$

Equivalently,

$$\mu_1 - \mu_2 = 0$$
$$\mu_2 - \mu_3 = 0$$
$$...$$
$$\mu_5 - \mu_6 = 0$$

We can test orthogonal polynomials for 4 equally spaced time points. To test for example the null hypothesis that quadratic and cubic effects are jointly equal to 0, we would define $\mathbf{C}$

$$\mathbf{C} = \begin{pmatrix} 1 & -1 & -1 & 1 \\ -1 & 3 & -3 & 1 \end{pmatrix}$$

25.0.3.2 Two-Sample Tests

Consider the analogous two sample multivariate tests.

Example: we have data on two independent random samples, one sample from each of two populations

$$\mathbf{y}_{1i} \sim N_p(\ _{\mathbf{1}},\ )$$
$$\mathbf{y}_{2j} \sim N_p(\ _{\mathbf{2}},\ )$$

We **assume**

- normality
- equal variance-covariance matrices
- independent random samples

We can summarize our data using the **sufficient statistics** $\bar{\mathbf{y}}_1, \mathbf{S}_1, \bar{\mathbf{y}}_2, \mathbf{S}_2$ with respective sample sizes, n_1, n_2

Since we assume that $_1 = \ _2 = \ $, compute a pooled estimate of the variance-covariance matrix on $n_1 + n_2 - 2$ df

$$\mathbf{S} = \frac{(n_1 - 1)\mathbf{S}_1 + (n_2 - 1)\mathbf{S}_2}{(n_1 - 1) + (n_2 - 1)}$$

$$H_0 : \ _1 = \ _2$$
$$H_a : \ _1 \neq \ _2$$

At least one element of the mean vectors is different

We use

- $\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2$ to estimate $\mu_1 - \mu_2$

- **S** to estimate

  Note: because we assume the two populations are independent, there is no covariance

  $cov(\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2) = var(\bar{\mathbf{y}}_1) + var(\bar{\mathbf{y}}_2) = \frac{1}{n_1} + \frac{2}{n_2} = (\frac{1}{n_1} + \frac{1}{n_2})$

Reject H_0 if

$$\begin{aligned} T^2 &= (\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2)'\{\mathbf{S}(\frac{1}{n_1} + \frac{1}{n_2})\}^{-1}(\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2) \\ &= \frac{n_1 n_2}{n_1 + n_2}(\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2)'\{\mathbf{S}\}^{-1}(\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2) \\ &\geq \frac{(n_1 + n_2 - 2)p}{n_1 + n_2 - p - 1} f_{(1-\alpha, n_1+n_2-p-1)} \end{aligned}$$

or equivalently, if

$$F = \frac{n_1 + n_2 - p - 1}{(n_1 + n_2 - 2)p} T^2 \geq f_{(1-\alpha, p, n_1+n_2-p-1)}$$

A $100(1-\alpha)\%$ confidence region for $\mu_1 - \mu_2$ consists of all vector δ which satisfy

$$\frac{n_1 n_2}{n_1 + n_2}(\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2 - \ )'\mathbf{S}^{-1}(\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2 - \ ) \leq \frac{(n_1 + n_2 - 2)p}{n_1 + n_2 - p - 1} f_{(1-\alpha, p, n_1+n_2-p-1)}$$

The simultaneous confidence intervals for all linear combinations of $\mu_1 - \mu_2$ have the form

$$\mathbf{a}'(\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2) \pm \sqrt{\frac{(n_1 + n_2 - 2)p}{n_1 + n_2 - p - 1} f_{(1-\alpha, p, n_1+n_2-p-1)}} \times \sqrt{\mathbf{a}'\mathbf{S}\mathbf{a}(\frac{1}{n_1} + \frac{1}{n_2})}$$

Bonferroni intervals, for k combinations

$$(\bar{y}_{1i} - \bar{y}_{2i}) \pm t_{(1-\alpha/2k, n_1+n_2-2)} \sqrt{(\frac{1}{n_1} + \frac{1}{n_2}) s_{ii}}$$

25.0.3.3 Model Assumptions

If model assumption are not met

- Unequal Covariance Matrices

  - If $n_1 = n_2$ (large samples) there is little effect on the Type I error rate and power fo the two sample test
  - If $n_1 > n_2$ and the eigenvalues of ${}_1{}_2^{-1}$ are less than 1, the Type I error level is inflated
  - If $n_1 > n_2$ and some eigenvalues of ${}_1{}_2^{-1}$ are greater than 1, the Type I error rate is too small, leading to a reduction in power

- Sample Not Normal
  - Type I error level of the two sample T^2 test isn't much affect by moderate departures from normality if the two populations being sampled have similar distributions
  - One sample T^2 test is much more sensitive to lack of normality, especially when the distribution is skewed.
  - Intuitively, you can think that in one sample your distribution will be sensitive, but the distribution of the difference between two similar distributions will not be as sensitive.
  - Solutions:
    * Transform to make the data more normal
    * Large large samples, use the χ^2 (Wald) test, in which populations don't need to be normal, or equal sample sizes, or equal variance-covariance matrices
      · $H_0 : \mu_1 - \mu_2 = 0$ use $(\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2)'(\frac{1}{n_1}\mathbf{S}_1 + \frac{1}{n_2}\mathbf{S}_2)^{-1}(\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2) \dot{\sim} \chi^2_{(p)}$

25.0.3.3.1 Equal Covariance Matrices Tests With independent random samples from k populations of p-dimensional vectors. We compute the sample covariance matrix for each, $\mathbf{S}_i$, where $i = 1, ..., k$

$$H_0 : {}_1 = {}_2 = ... = {}_k =$$
$$H_a : \text{at least 2 are different}$$

Assume H_0 is true, we would use a pooled estimate of the common covariance matrix,

$$\mathbf{S} = \frac{\sum_{i=1}^k (n_i - 1)\mathbf{S}_i}{\sum_{i=1}^k (n_i - 1)}$$

with $\sum_{i=1}^k (n_i - 1)$

25.0.3.3.1.1 Bartlett's Test (a modification of the likelihood ratio test). Define

$$N = \sum_{i=1}^{k} n_i$$

and (note: || are determinants here, not absolute value)

$$M = (N-k)\log|\mathbf{S}| - \sum_{i=1}^{k}(n_i - 1)\log|\mathbf{S}_i|$$

$$C^{-1} = 1 - \frac{2p^2+3p-1}{6(p+1)(k-1)}\{\sum_{i=1}^{k}(\frac{1}{n_i-1}) - \frac{1}{N-k}\}$$

- Reject H_0 when $MC^{-1} > \chi^2_{1-\alpha,(k-1)p(p+1)/2}$
- If not all samples are from normal populations, MC^{-1} has a distribution which is often shifted to the right of the nominal χ^2 distribution, which means H_0 is often rejected even when it is true (the Type I error level is inflated). Hence, it is better to test individual normality first, or then multivariate normality before you do Bartlett's test.

25.0.3.4 Two-Sample Repeated Measurements

- Define $\mathbf{y}_{hi} = (y_{hi1}, ..., y_{hit})'$ to be the observations from the i-th subject in the h-th group for times 1 through T
- Assume that $\mathbf{y}_{11}, ..., \mathbf{y}_{1n_1}$ are iid $N_t(\ _1,\ )$ and that $\mathbf{y}_{21}, ..., \mathbf{y}_{2n_2}$ are iid $N_t(\ _2,\ )$
- $H_0 : \mathbf{C}(\ _1 -\ _2) = \mathbf{0}_c$ where $\mathbf{C}$ is a $c \times t$ matrix of rank c where $c \leq t$
- The test statistic has the form

$$T^2 = \frac{n_1 n_2}{n_1+n_2}(\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2)'\mathbf{C}'(\mathbf{CSC}')^{-1}\mathbf{C}(\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2)$$

where $\mathbf{S}$ is the pooled covariance estimate. Then,

$$F = \frac{n_1+n_2-c-1}{(n_1+n_2-2)c}T^2 \sim f_{(c,n_1+n_2-c-1)}$$

when H_0 is true

If the null hypothesis $H_0 : \mu_1 = \mu_2$ is rejected. A weaker hypothesis is that the profiles for the two groups are parallel.

$$\mu_{11} - \mu_{21} = \mu_{12} - \mu_{22}$$
$$\vdots$$
$$\mu_{1t-1} - \mu_{2t-1} = \mu_{1t} - \mu_{2t}$$

The null hypothesis matrix term is then

$H_0 : \mathbf{C}(\mu_1 - \mu_2) = \mathbf{0}_c$, where $c = t - 1$ and

$$\mathbf{C} = \begin{pmatrix} 1 & -1 & 0 & \dots & 0 \\ 0 & 1 & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -1 \end{pmatrix}_{(t-1)\times t}$$

```
# One-sample Hotelling's T^2 test
#  Create data frame
plants <- data.frame(
    y1 = c(2.11, 2.36, 2.13, 2.78, 2.17),
    y2 = c(10.1, 35.0, 2.0, 6.0, 2.0),
    y3 = c(3.4, 4.1, 1.9, 3.8, 1.7)
)

# Center the data with
# the hypothesized means and make a matrix
plants_ctr <- plants %>%
    transmute(y1_ctr = y1 - 2.85,
              y2_ctr = y2 - 15.0,
              y3_ctr = y3 - 6.0) %>%
    as.matrix()

# Use anova.mlm to calculate Wilks' lambda
onesamp_fit <- anova(lm(plants_ctr ~ 1), test = "Wilks")
onesamp_fit
#> Analysis of Variance Table
#>
#>             Df    Wilks approx F num Df den Df  Pr(>F)
#> (Intercept)  1 0.054219   11.629      3      2 0.08022 .
#> Residuals    4
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

can't reject the null of hypothesized vector of means

```
# Paired-Sample Hotelling's T^2 test
library(ICSNP)
```

```r
#  Create data frame
waste <- data.frame(
    case = 1:11,
    com_y1 = c(6, 6, 18, 8, 11, 34, 28, 71, 43, 33, 20),
    com_y2 = c(27, 23, 64, 44, 30, 75, 26, 124, 54, 30, 14),
    state_y1 = c(25, 28, 36, 35, 15, 44, 42, 54, 34, 29, 39),
    state_y2 = c(15, 13, 22, 29, 31, 64, 30, 64, 56, 20, 21)
)

# Calculate the difference between commercial and state labs
waste_diff <- waste %>%
    transmute(y1_diff = com_y1 - state_y1,
              y2_diff = com_y2 - state_y2)
# Run the test
paired_fit <- HotellingsT2(waste_diff)
# value T.2 in the output corresponds to
# the approximate F-value in the output from anova.mlm
paired_fit
#>
#>  Hotelling's one sample T2-test
#>
#> data:  waste_diff
#> T.2 = 6.1377, df1 = 2, df2 = 9, p-value = 0.02083
#> alternative hypothesis: true location is not equal to c(0,0)
```

reject the null that the two labs' measurements are equal

```r
# Independent-Sample Hotelling's T^2 test with Bartlett's test

# Read in data
steel <- read.table("images/steel.dat")
names(steel) <- c("Temp", "Yield", "Strength")
str(steel)
#> 'data.frame':    12 obs. of  3 variables:
#>  $ Temp    : int  1 1 1 1 1 2 2 2 2 2 ...
#>  $ Yield   : int  33 36 35 38 40 35 36 38 39 41 ...
#>  $ Strength: int  60 61 64 63 65 57 59 59 61 63 ...

# Plot the data
ggplot(steel, aes(x = Yield, y = Strength)) +
    geom_text(aes(label = Temp), size = 5) +
    geom_segment(aes(
        x = 33,
        y = 57.5,
        xend = 42,
        yend = 65
```

```
    ), col = "red")
```

```
# Bartlett's test for equality of covariance matrices
# same thing as Box's M test in the multivariate setting
bart_test <- boxM(steel[, -1], steel$Temp)
bart_test # fail to reject the null of equal covariances
#>
#>  Box's M-test for Homogeneity of Covariance Matrices
#>
#> data:  steel[, -1]
#> Chi-Sq (approx.) = 0.38077, df = 3, p-value = 0.9442

# anova.mlm
twosamp_fit <-
    anova(lm(cbind(Yield, Strength) ~ factor(Temp),
             data = steel),
          test = "Wilks")
twosamp_fit
#> Analysis of Variance Table
#>
#>               Df    Wilks approx F num Df den Df    Pr(>F)
#> (Intercept)    1 0.001177   3818.1      2      9 6.589e-14 ***
#> factor(Temp)   1 0.294883     10.8      2      9  0.004106 **
#> Residuals     10
#> ---
```

```
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# ICSNP package
twosamp_fit2 <-
    HotellingsT2(cbind(steel$Yield, steel$Strength) ~
                     factor(steel$Temp))
twosamp_fit2
#>
#>  Hotelling's two sample T2-test
#>
#> data:  cbind(steel$Yield, steel$Strength) by factor(steel$Temp)
#> T.2 = 10.76, df1 = 2, df2 = 9, p-value = 0.004106
#> alternative hypothesis: true location difference is not equal to c(0,0)
```

reject null. Hence, there is a difference in the means of the bivariate normal distributions

25.1 MANOVA

Multivariate Analysis of Variance

One-way MANOVA

Compare treatment means for h different populations

Population 1: $\mathbf{y}_{11}, \mathbf{y}_{12}, \dots, \mathbf{y}_{1n_1} \sim iddN_p(\ _1,\ )$

$\vdots$

Population h: $\mathbf{y}_{h1}, \mathbf{y}_{h2}, \dots, \mathbf{y}_{hn_h} \sim iddN_p(\ _h,\ )$

Assumptions

1. Independent random samples from h different populations
2. Common covariance matrices
3. Each population is multivariate **normal**

Calculate the summary statistics $\bar{\mathbf{y}}_i, \mathbf{S}$ and the pooled estimate of the covariance matrix $\mathbf{S}$

Similar to the univariate one-way ANVOA, we can use the effects model formulation $\ _i = \ + \ _i$, where

- $\ _i$ is the population mean for population i
- is the overall mean effect
- $\ _i$ is the treatment effect of the i-th treatment.

For the one-way model: $\mathbf{y}_{ij} = \mu + \tau_i + \epsilon_{ij}$ for $i = 1, .., h; j = 1, ..., n_i$ and $\epsilon_{ij} \sim N_p(\mathbf{0},\ )$

However, the above model is over-parameterized (i.e., infinite number of ways to define and the $_i$'s such that they add up to μ_i. Thus we can constrain by having

$$\sum_{i=1}^{h} n_i \tau_i = 0$$

or

$$_h = 0$$

The observational equivalent of the effects model is

$$\begin{aligned}\mathbf{y}_{ij} &= \bar{\mathbf{y}} + (\bar{\mathbf{y}}_i - \bar{\mathbf{y}}) + (\mathbf{y}_{ij} - \bar{\mathbf{y}}_i) \\ &= \text{overall sample mean} + \text{treatement effect} + \text{residual (under univariate ANOVA)}\end{aligned}$$

After manipulation

$$\sum_{i=1}^{h}\sum_{j=1}^{n_i}(\bar{\mathbf{y}}_{ij} - \bar{\mathbf{y}})(\bar{\mathbf{y}}_{ij} - \bar{\mathbf{y}})' = \sum_{i=1}^{h} n_i(\bar{\mathbf{y}}_i - \bar{\mathbf{y}})(\bar{\mathbf{y}}_i - \bar{\mathbf{y}})' + \sum_{i=1}^{h}\sum_{j=1}^{n_i}(\bar{\mathbf{y}}_{ij} - \bar{\mathbf{y}})(\bar{\mathbf{y}}_{ij} - \bar{\mathbf{y}}_i)'$$

LHS = Total corrected sums of squares and cross products (SSCP) matrix

RHS =

- 1st term = Treatment (or between subjects) sum of squares and cross product matrix (denoted H;B)
- 2nd term = residual (or within subject) SSCP matrix denoted (E;W)

Note:

$$\mathbf{E} = (n_1 - 1)\mathbf{S}_1 + ... + (n_h - 1)\mathbf{S}_h = (n - h)\mathbf{S}$$

MANOVA table

Table 25.1: MONOVA table

Source	SSCP	df
Treatment	$\mathbf{H}$	$h - 1$
Residual (error)	$\mathbf{E}$	$\sum_{i=1}^{h} n_i - h$

Source	SSCP	df
Total Corrected	$\mathbf{H}+\mathbf{E}$	$\sum_{i=1}^{h} n_i - 1$

$$H_0 : \tau_1 = \tau_2 = \cdots = \tau_h = \mathbf{0}$$

We consider the relative "sizes" of $\mathbf{E}$ and $\mathbf{H}+\mathbf{E}$

Wilk's Lambda

Define Wilk's Lambda

$$\Lambda^* = \frac{|\mathbf{E}|}{|\mathbf{H}+\mathbf{E}|}$$

Properties:

1. Wilk's Lambda is equivalent to the F-statistic in the univariate case
2. The exact distribution of Λ^* can be determined for especial cases.
3. For large sample sizes, reject H_0 if

$$-(\sum_{i=1}^{h} n_i - 1 - \frac{p+h}{2}) \log(\Lambda^*) > \chi^2_{(1-\alpha, p(h-1))}$$

25.1.1 Testing General Hypotheses

- h different treatments
- with the i-th treatment
- applied to n_i subjects that
- are observed for p repeated measures.

Consider this a p dimensional obs on a random sample from each of h different treatment populations.

$$\mathbf{y}_{ij} = \ + \ _i + \ _{ij}$$

for $i = 1, .., h$ and $j = 1, .., n_i$

Equivalently,

$$\mathbf{Y} = \mathbf{XB} +$$

where $n = \sum_{i=1}^{h} n_i$ and with restriction $_h = 0$

$$\mathbf{Y}_{(n\times p)} = \begin{bmatrix} \mathbf{y}'_{11} \\ \vdots \\ \mathbf{y}'_{1n_1} \\ \vdots \\ \mathbf{y}'_{hn_h} \end{bmatrix}, \mathbf{B}_{(h\times p)} = \begin{bmatrix} {}' \\ {}' \\ {}_1 \\ \vdots \\ {}'_{h-1} \end{bmatrix}, \ {}_{(n\times p)} = \begin{bmatrix} \epsilon'_{11} \\ \vdots \\ \epsilon'_{1n_1} \\ \vdots \\ \epsilon'_{hn_h} \end{bmatrix}$$

$$\mathbf{X}_{(n\times h)} = \begin{bmatrix} 1 & 1 & 0 & ... & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 1 & 0 & ... & 0 \\ \vdots & \vdots & \vdots & ... & \vdots \\ 1 & 0 & 0 & ... & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 0 & 0 & ... & 0 \end{bmatrix}$$

Estimation

$$\hat{\mathbf{B}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$$

Rows of $\mathbf{Y}$ are independent (i.e., $var(\mathbf{Y}) = \mathbf{I}_n \otimes$, an $np \times np$ matrix, where $\otimes$ is the Kronecker product).

$$H_0 : \mathbf{LBM} = 0$$
$$H_a : \mathbf{LBM} \neq 0$$

where

- $\mathbf{L}$ is a $g \times h$ matrix of full row rank ($g \leq h$) = comparisons across groups
- $\mathbf{M}$ is a $p \times u$ matrix of full column rank ($u \leq p$) = comparisons across traits

The general treatment corrected sums of squares and cross product is

$$\mathbf{H} = \mathbf{M}'\mathbf{Y}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}'[\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}']^{-1}\mathbf{L}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}\mathbf{M}$$

or for the null hypothesis $H_0 : \mathbf{LBM} = \mathbf{D}$

$$\mathbf{H} = (\mathbf{L}\hat{\mathbf{B}}\mathbf{M} - \mathbf{D})'[\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{L}]^{-1}(\mathbf{L}\hat{\mathbf{B}}\mathbf{M} - \mathbf{D})$$

The general matrix of residual sums of squares and cross product

$$\mathbf{E} = \mathbf{M}'\mathbf{Y}'[\mathbf{I} - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}']\mathbf{Y}\mathbf{M} = \mathbf{M}'[\mathbf{Y}'\mathbf{Y} - \hat{\mathbf{B}}'(\mathbf{X}'\mathbf{X})^{-1}\hat{\mathbf{B}}]\mathbf{M}$$

We can compute the following statistic eigenvalues of $\mathbf{HE}^{-1}$

- Wilk's Criterion: $\Lambda^* = \frac{|\mathbf{E}|}{|\mathbf{H}+\mathbf{E}|}$. The df depend on the rank of $\mathbf{L}, \mathbf{M}, \mathbf{X}$
- Lawley-Hotelling Trace: $U = tr(\mathbf{HE}^{-1})$
- Pillai Trace: $V = tr(\mathbf{H}(\mathbf{H} + \mathbf{E}^{-1})$
- Roy's Maximum Root: largest eigenvalue of $\mathbf{HE}^{-1}$

If H_0 is true and n is large, $-(n - 1 - \frac{p+h}{2}) \ln \Lambda^* \sim \chi^2_{p(h-1)}$. Some special values of p and h can give exact F-dist under H_0

```r
# One-way MANOVA

library(car)
library(emmeans)
library(profileR)
library(tidyverse)

## Read in the data
gpagmat <- read.table("images/gpagmat.dat")

## Change the variable names
names(gpagmat) <- c("y1", "y2", "admit")

## Check the structure
str(gpagmat)
#> 'data.frame':    85 obs. of  3 variables:
#>  $ y1   : num  2.96 3.14 3.22 3.29 3.69 3.46 3.03 3.19 3.63 3.59 ...
#>  $ y2   : int  596 473 482 527 505 693 626 663 447 588 ...
#>  $ admit: int  1 1 1 1 1 1 1 1 1 1 ...


## Plot the data
gg <- ggplot(gpagmat, aes(x = y1, y = y2)) +
    geom_text(aes(label = admit, col = as.character(admit))) +
    scale_color_discrete(name = "Admission",
                         labels = c("Admit", "Do not admit", "Borderline")) +
    scale_x_continuous(name = "GPA") +
    scale_y_continuous(name = "GMAT")

## Fit one-way MANOVA
oneway_fit <- manova(cbind(y1, y2) ~ admit, data = gpagmat)
summary(oneway_fit, test = "Wilks")
#>           Df  Wilks approx F num Df den Df    Pr(>F)
#> admit      1 0.6126   25.927      2     82 1.881e-09 ***
#> Residuals 83
```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

reject the null of equal multivariate mean vectors between the three admmission groups

```
# Repeated Measures MANOVA


## Create data frame
stress <- data.frame(
    subject = 1:8,
    begin = c(3, 2, 5, 6, 1, 5, 1, 5),
    middle = c(3, 4, 3, 7, 4, 7, 1, 2),
    final = c(6, 7, 4, 7, 6, 7, 3, 5)
)
```

- If independent = time with 3 levels -> univariate ANOVA (require sphericity assumption (i.e., the variances for all differences are equal))
- If each level of independent time as a separate variable -> MANOVA (does not require sphericity assumption)

```
## MANOVA
stress_mod <- lm(cbind(begin, middle, final) ~ 1, data = stress)
idata <-
    data.frame(time = factor(
        c("begin", "middle", "final"),
        levels = c("begin", "middle", "final")
    ))
repeat_fit <-
    Anova(
        stress_mod,
        idata = idata,
        idesign = ~ time,
        icontrasts = "contr.poly"
    )
summary(repeat_fit)
#>
#> Type III Repeated Measures MANOVA Tests:
#>
#> ------------------------------------------
#>
#> Term: (Intercept)
#>
#>  Response transformation matrix:
#>        (Intercept)
#> begin             1
```

```
#> middle           1
#> final            1
#>
#> Sum of squares and products for the hypothesis:
#>             (Intercept)
#> (Intercept)        1352
#>
#> Multivariate Tests: (Intercept)
#>                  Df test stat approx F num Df den Df     Pr(>F)
#> Pillai            1  0.896552 60.66667      1      7 0.00010808 ***
#> Wilks             1  0.103448 60.66667      1      7 0.00010808 ***
#> Hotelling-Lawley  1  8.666667 60.66667      1      7 0.00010808 ***
#> Roy               1  8.666667 60.66667      1      7 0.00010808 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> ------------------------------------------
#>
#> Term: time
#>
#>  Response transformation matrix:
#>               time.L     time.Q
#> begin  -7.071068e-01  0.4082483
#> middle -7.850462e-17 -0.8164966
#> final   7.071068e-01  0.4082483
#>
#> Sum of squares and products for the hypothesis:
#>           time.L   time.Q
#> time.L 18.062500 6.747781
#> time.Q  6.747781 2.520833
#>
#> Multivariate Tests: time
#>                  Df test stat approx F num Df den Df   Pr(>F)
#> Pillai            1 0.7080717 7.276498      2      6 0.024879 *
#> Wilks             1 0.2919283 7.276498      2      6 0.024879 *
#> Hotelling-Lawley  1 2.4254992 7.276498      2      6 0.024879 *
#> Roy               1 2.4254992 7.276498      2      6 0.024879 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Univariate Type III Repeated-Measures ANOVA Assuming Sphericity
#>
#>             Sum Sq num Df Error SS den Df F value    Pr(>F)
#> (Intercept) 450.67      1    52.00      7 60.6667 0.0001081 ***
#> time         20.58      2    24.75     14  5.8215 0.0144578 *
```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>
#> Mauchly Tests for Sphericity
#>
#>      Test statistic p-value
#> time         0.7085 0.35565
#>
#>
#> Greenhouse-Geisser and Huynh-Feldt Corrections
#>  for Departure from Sphericity
#>
#>       GG eps Pr(>F[GG])
#> time 0.77429    0.02439 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>         HF eps Pr(>F[HF])
#> time 0.9528433 0.01611634
```

can't reject the null hypothesis of sphericity, hence univariate ANOVA is also appropriate.We also see linear significant time effect, but no quadratic time effect

```
## Polynomial contrasts
# What is the reference for the marginal means?
ref_grid(stress_mod, mult.name = "time")
#> 'emmGrid' object with variables:
#>     1 = 1
#>     time = multivariate response levels: begin, middle, final

# marginal means for the levels of time
contr_means <- emmeans(stress_mod, ~ time, mult.name = "time")
contrast(contr_means, method = "poly")
#>  contrast  estimate    SE df t.ratio p.value
#>  linear        2.12 0.766  7   2.773  0.0276
#>  quadratic     1.38 0.944  7   1.457  0.1885
```

```
# MANOVA



## Read in Data
heart <- read.table("images/heart.dat")
names(heart) <- c("drug", "y1", "y2", "y3", "y4")
## Create a subject ID nested within drug
```

```
heart <- heart %>%
    group_by(drug) %>%
    mutate(subject = row_number()) %>%
    ungroup()
str(heart)
#> tibble [24 x 6] (S3: tbl_df/tbl/data.frame)
#>  $ drug   : chr [1:24] "ax23" "ax23" "ax23" "ax23" ...
#>  $ y1     : int [1:24] 72 78 71 72 66 74 62 69 85 82 ...
#>  $ y2     : int [1:24] 86 83 82 83 79 83 73 75 86 86 ...
#>  $ y3     : int [1:24] 81 88 81 83 77 84 78 76 83 80 ...
#>  $ y4     : int [1:24] 77 82 75 69 66 77 70 70 80 84 ...
#>  $ subject: int [1:24] 1 2 3 4 5 6 7 8 1 2 ...

## Create means summary for profile plot,
# pivot longer for plotting with ggplot
heart_means <- heart %>%
    group_by(drug) %>%
    summarize_at(vars(starts_with("y")), mean) %>%
    ungroup() %>%
    pivot_longer(-drug, names_to = "time", values_to = "mean") %>%
    mutate(time = as.numeric(as.factor(time)))
gg_profile <- ggplot(heart_means, aes(x = time, y = mean)) +
    geom_line(aes(col = drug)) +
    geom_point(aes(col = drug)) +
    ggtitle("Profile Plot") +
    scale_y_continuous(name = "Response") +
    scale_x_discrete(name = "Time")
gg_profile
```

```
## Fit model
heart_mod <- lm(cbind(y1, y2, y3, y4) ~ drug, data = heart)
man_fit <- car::Anova(heart_mod)
summary(man_fit)
#>
#> Type II MANOVA Tests:
#>
#> Sum of squares and products for error:
#>         y1       y2       y3      y4
#> y1 641.00 601.750 535.250 426.00
#> y2 601.75 823.875 615.500 534.25
#> y3 535.25 615.500 655.875 555.25
#> y4 426.00 534.250 555.250 674.50
#>
#> ------------------------------------------
#>
#> Term: drug
#>
#> Sum of squares and products for the hypothesis:
#>         y1        y2        y3    y4
#> y1 567.00 335.2500  42.7500 387.0
#> y2 335.25 569.0833 404.5417 367.5
#> y3  42.75 404.5417 391.0833 171.0
#> y4 387.00 367.5000 171.0000 316.0
#>
#> Multivariate Tests: drug
#>                  Df test stat  approx F num Df den Df     Pr(>F)
```

```
#> Pillai            2  1.283456  8.508082      8     38 1.5010e-06 ***
#> Wilks             2  0.079007 11.509581      8     36 6.3081e-08 ***
#> Hotelling-Lawley  2  7.069384 15.022441      8     34 3.9048e-09 ***
#> Roy               2  6.346509 30.145916      4     19 5.4493e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

reject the null hypothesis of no difference in means between treatments

```
## Contrasts
heart$drug <- factor(heart$drug)
L <- matrix(c(0, 2,
              1, -1,-1, -1), nrow = 3, byrow = T)
colnames(L) <- c("bww9:ctrl", "ax23:rest")
rownames(L) <- unique(heart$drug)
contrasts(heart$drug) <- L
contrasts(heart$drug)
#>      bww9:ctrl ax23:rest
#> ax23         0         2
#> bww9         1        -1
#> ctrl        -1        -1

# do not set contrast L if you do further analysis (e.g., Anova, lm)
# do M matrix instead

M <- matrix(c(1, -1, 0, 0,
              0, 1, -1, 0,
              0, 0, 1, -1), nrow = 4)
## update model to test contrasts
heart_mod2 <- update(heart_mod)
coef(heart_mod2)
#>                  y1         y2        y3    y4
#> (Intercept)   75.00 78.9583333 77.041667 74.75
#> drugbww9:ctrl  4.50  5.8125000  3.562500  4.25
#> drugax23:rest -2.25  0.7708333  1.979167 -0.75

# Hypothesis test for bww9 vs control after transformation M
# same as linearHypothesis(heart_mod, hypothesis.matrix = c(0,1,-1), P = M)
bww9vctrl <-
    car::linearHypothesis(heart_mod2,
                          hypothesis.matrix = c(0, 1, 0),
                          P = M)
bww9vctrl
#>
#>  Response transformation matrix:
#>    [,1] [,2] [,3]
```

```
#> y1    1    0    0
#> y2   -1    1    0
#> y3    0   -1    1
#> y4    0    0   -1
#>
#> Sum of squares and products for the hypothesis:
#>          [,1]   [,2]     [,3]
#> [1,]  27.5625 -47.25  14.4375
#> [2,] -47.2500  81.00 -24.7500
#> [3,]  14.4375 -24.75   7.5625
#>
#> Sum of squares and products for error:
#>          [,1]     [,2]    [,3]
#> [1,]  261.375 -141.875  28.000
#> [2,] -141.875  248.750 -19.375
#> [3,]   28.000  -19.375 219.875
#>
#> Multivariate Tests:
#>                  Df test stat approx F num Df den Df Pr(>F)
#> Pillai            1 0.2564306 2.184141      3     19 0.1233
#> Wilks             1 0.7435694 2.184141      3     19 0.1233
#> Hotelling-Lawley  1 0.3448644 2.184141      3     19 0.1233
#> Roy               1 0.3448644 2.184141      3     19 0.1233

bww9vctrl <-
    car::linearHypothesis(heart_mod,
                          hypothesis.matrix = c(0, 1, -1),
                          P = M)
bww9vctrl
#>
#>  Response transformation matrix:
#>    [,1] [,2] [,3]
#> y1    1    0    0
#> y2   -1    1    0
#> y3    0   -1    1
#> y4    0    0   -1
#>
#> Sum of squares and products for the hypothesis:
#>          [,1]   [,2]     [,3]
#> [1,]  27.5625 -47.25  14.4375
#> [2,] -47.2500  81.00 -24.7500
#> [3,]  14.4375 -24.75   7.5625
#>
#> Sum of squares and products for error:
#>          [,1]     [,2]    [,3]
```

```
#> [1,]  261.375 -141.875  28.000
#> [2,] -141.875  248.750 -19.375
#> [3,]   28.000  -19.375 219.875
#>
#> Multivariate Tests:
#>                  Df test stat approx F num Df den Df Pr(>F)
#> Pillai            1 0.2564306 2.184141      3     19 0.1233
#> Wilks             1 0.7435694 2.184141      3     19 0.1233
#> Hotelling-Lawley  1 0.3448644 2.184141      3     19 0.1233
#> Roy               1 0.3448644 2.184141      3     19 0.1233
```

there is no significant difference in means between the control and `bww9` drug

```
# Hypothesis test for ax23 vs rest after transformation M
axx23vrest <-
    car::linearHypothesis(heart_mod2,
                          hypothesis.matrix = c(0, 0, 1),
                          P = M)
axx23vrest
#>
#>  Response transformation matrix:
#>    [,1] [,2] [,3]
#> y1    1    0    0
#> y2   -1    1    0
#> y3    0   -1    1
#> y4    0    0   -1
#>
#> Sum of squares and products for the hypothesis:
#>           [,1]       [,2]      [,3]
#> [1,]  438.0208  175.20833 -395.7292
#> [2,]  175.2083   70.08333 -158.2917
#> [3,] -395.7292 -158.29167  357.5208
#>
#> Sum of squares and products for error:
#>          [,1]     [,2]    [,3]
#> [1,]  261.375 -141.875  28.000
#> [2,] -141.875  248.750 -19.375
#> [3,]   28.000  -19.375 219.875
#>
#> Multivariate Tests:
#>                  Df test stat approx F num Df den Df     Pr(>F)
#> Pillai            1  0.855364 37.45483      3     19 3.5484e-08 ***
#> Wilks             1  0.144636 37.45483      3     19 3.5484e-08 ***
#> Hotelling-Lawley  1  5.913921 37.45483      3     19 3.5484e-08 ***
#> Roy               1  5.913921 37.45483      3     19 3.5484e-08 ***
#> ---
```

```
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

axx23vrest <-
    car::linearHypothesis(heart_mod,
                          hypothesis.matrix = c(2, -1, 1),
                          P = M)
axx23vrest
#>
#>  Response transformation matrix:
#>    [,1] [,2] [,3]
#> y1    1    0    0
#> y2   -1    1    0
#> y3    0   -1    1
#> y4    0    0   -1
#>
#> Sum of squares and products for the hypothesis:
#>           [,1]       [,2]      [,3]
#> [1,]  402.5208  127.41667 -390.9375
#> [2,]  127.4167   40.33333 -123.7500
#> [3,] -390.9375 -123.75000  379.6875
#>
#> Sum of squares and products for error:
#>          [,1]     [,2]    [,3]
#> [1,]  261.375 -141.875  28.000
#> [2,] -141.875  248.750 -19.375
#> [3,]   28.000  -19.375 219.875
#>
#> Multivariate Tests:
#>                  Df test stat approx F num Df den Df     Pr(>F)
#> Pillai            1  0.842450 33.86563      3     19 7.9422e-08 ***
#> Wilks             1  0.157550 33.86563      3     19 7.9422e-08 ***
#> Hotelling-Lawley  1  5.347205 33.86563      3     19 7.9422e-08 ***
#> Roy               1  5.347205 33.86563      3     19 7.9422e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

there is a significant difference in means between ax23 drug treatment and the rest of the treatments

25.1.2 Profile Analysis

Examine similarities between the treatment effects (between subjects), which is useful for longitudinal analysis. Null is that all treatments have the same average effect.

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_h$$

Equivalently,

$$H_0 : \tau_1 = \tau_2 = \cdots = \tau_h$$

The exact nature of the similarities and differences between the treatments can be examined under this analysis.

Sequential steps in profile analysis:

1. Are the profiles **parallel**? (i.e., is there no interaction between treatment and time)
2. Are the profiles **coincidental**? (i.e., are the profiles identical?)
3. Are the profiles **horizontal**? (i.e., are there no differences between any time points?)

If we reject the null hypothesis that the profiles are parallel, we can test

- Are there differences among groups within some subset of the total time points?
- Are there differences among time points in a particular group (or groups)?
- Are there differences within some subset of the total time points in a particular group (or groups)?

Example

- 4 times (p = 4)
- 3 treatments (h=3)

25.1.2.1 Parallel Profile

Are the profiles for each population identical expect for a mean shift?

$$\begin{aligned} H_0 : \mu_{11} - \mu_{21} - \mu_{12} - \mu_{22} &= \cdots = \mu_{1t} - \mu_{2t} \\ \mu_{11} - \mu_{31} - \mu_{12} - \mu_{32} &= \cdots = \mu_{1t} - \mu_{3t} \\ &\cdots \end{aligned}$$

for $h - 1$ equations

Equivalently,

$$H_0 : \mathbf{LBM} = \mathbf{0}$$

$$\mathbf{LBM} = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \mu_{11} & \cdots & \mu_{14} \\ \mu_{21} & \cdots & \mu_{24} \\ \mu_{31} & \cdots & \mu_{34} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} = \mathbf{0}$$

where this is the cell means parameterization of $\mathbf{B}$

The multiplication of the first 2 matrices $\mathbf{LB}$ is

$$\begin{bmatrix} \mu_{11} - \mu_{21} & \mu_{12} - \mu_{22} & \mu_{13} - \mu_{23} & \mu_{14} - \mu_{24} \\ \mu_{11} - \mu_{31} & \mu_{12} - \mu_{32} & \mu_{13} - \mu_{33} & \mu_{14} - \mu_{34} \end{bmatrix}$$

which is the differences in treatment means at the same time

Multiplying by $\mathbf{M}$, we get the comparison across time

$$\begin{bmatrix} (\mu_{11} - \mu_{21}) - (\mu_{12} - \mu_{22}) & (\mu_{11} - \mu_{21}) - (\mu_{13} - \mu_{23}) & (\mu_{11} - \mu_{21}) - (\mu_{14} - \mu_{24}) \\ (\mu_{11} - \mu_{31}) - (\mu_{12} - \mu_{32}) & (\mu_{11} - \mu_{31}) - (\mu_{13} - \mu_{33}) & (\mu_{11} - \mu_{31}) - (\mu_{14} - \mu_{34}) \end{bmatrix}$$

Alternatively, we can also use the effects parameterization

$$\mathbf{LBM} = \begin{bmatrix} 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \mu' \\ \tau_1' \\ \tau_2' \\ \tau_3' \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} = \mathbf{0}$$

In both parameterizations, $rank(\mathbf{L}) = h - 1$ and $rank(\mathbf{M}) = p - 1$

We could also choose $\mathbf{L}$ and $\mathbf{M}$ in other forms

$$\mathbf{L} = \begin{bmatrix} 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

and

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix}$$

and still obtain the same result.

25.1.2.2 Coincidental Profiles

After we have evidence that the profiles are parallel (i.e., fail to reject the parallel profile test), we can ask whether they are identical?

Given profiles are **parallel**, then if the sums of the components of μ_i are identical for all the treatments, then the profiles are **identical**.

$$H_0 : \mathbf{1}'_p \mu_1 = \mathbf{1}'_p \mu_2 = \cdots = \mathbf{1}'_p \mu_h$$

Equivalently,

$$H_0 : \mathbf{LBM} = \mathbf{0}$$

where for the cell means parameterization

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix}$$

and

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}'$$

multiplication yields

$$\begin{bmatrix} (\mu_{11} + \mu_{12} + \mu_{13} + \mu_{14}) - (\mu_{31} + \mu_{32} + \mu_{33} + \mu_{34}) \\ (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24}) - (\mu_{31} + \mu_{32} + \mu_{33} + \mu_{34}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Different choices of $\mathbf{L}$ and $\mathbf{M}$ can yield the same result

25.1.2.3 Horizontal Profiles

Given that we can't reject the null hypothesis that all h profiles are the same, we can ask whether all of the elements of the common profile equal? (i.e., horizontal)

$$H_0 : \mathbf{LBM} = \mathbf{0}$$

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

and

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix}$$

hence,

$$\begin{bmatrix} (\mu_{11} - \mu_{12}) & (\mu_{12} - \mu_{13}) & (\mu_{13} + \mu_{14}) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

Note:

- If we fail to reject all 3 hypotheses, then we fail to reject the null hypotheses of both no difference between treatments and no differences between traits.

Test	Equivalent test for
Parallel profile	Interaction
Coincidental profile	main effect of between-subjects factor
Horizontal profile	main effect of repeated measures factor

```r
profile_fit <-
    pbg(
        data = as.matrix(heart[, 2:5]),
        group = as.matrix(heart[, 1]),
        original.names = TRUE,
        profile.plot = FALSE
    )
summary(profile_fit)
#> Call:
#> pbg(data = as.matrix(heart[, 2:5]), group = as.matrix(heart[,
#>     1]), original.names = TRUE, profile.plot = FALSE)
#>
#> Hypothesis Tests:
#> $`Ho: Profiles are parallel`
#>   Multivariate.Test Statistic  Approx.F num.df den.df      p.value
#> 1             Wilks 0.1102861 12.737599      6     38 7.891497e-08
#> 2            Pillai 1.0891707  7.972007      6     40 1.092397e-05
#> 3  Hotelling-Lawley 6.2587852 18.776356      6     36 9.258571e-10
#> 4               Roy 5.9550887 39.700592      3     20 1.302458e-08
#>
#> $`Ho: Profiles have equal levels`
#>             Df Sum Sq Mean Sq F value  Pr(>F)
#> group        2  328.7  164.35   5.918 0.00915 **
#> Residuals   21  583.2   27.77
#> ---
```

```
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> $`Ho: Profiles are flat`
#>          F df1 df2      p-value
#> 1 14.30928   3  19 4.096803e-05
# reject null hypothesis of parallel profiles
# reject the null hypothesis of coincidental profiles
# reject the null hypothesis that the profiles are flat
```

25.1.3 Summary

25.2 Principal Components

- Unsupervised learning
- find important features
- reduce the dimensions of the data set
- "decorrelate" multivariate vectors that have dependence.
- uses eigenvector/eigvenvalue decomposition of covariance (correlation) matrices.

According to the "spectral decomposition theorem", if $_{p\times p}$ i s a positive semi-definite, symmetric, real matrix, then there exists an orthogonal matrix $\mathbf{A}$ such

that $\mathbf{A}'\ \mathbf{A} = \Lambda$ where Λ is a diagonal matrix containing the eigenvalues

$$= \begin{pmatrix} \lambda_1 & 0 & ... & 0 \\ 0 & \lambda_2 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & \lambda_p \end{pmatrix}$$

$$\mathbf{A} = \begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 & ... & \mathbf{a}_p \end{pmatrix}$$

the i-th column of $\mathbf{A}$, $\mathbf{a}_i$, is the i-th $p \times 1$ eigenvector of that corresponds to the eigenvalue, λ_i , where $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p$. Alternatively, express in matrix decomposition:

$$= \mathbf{A}\ \mathbf{A}'$$

$$= \mathbf{A} \begin{pmatrix} \lambda_1 & 0 & ... & 0 \\ 0 & \lambda_2 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & \lambda_p \end{pmatrix} \mathbf{A}' = \sum_{i=1}^{p} \lambda_i \mathbf{a}_i \mathbf{a}_i'$$

where the outer product $\mathbf{a}_i \mathbf{a}_i'$ is a $p \times p$ matrix of rank 1.

For example,

$\mathbf{x} \sim N_2(\ , \ )$

$$= \begin{pmatrix} 5 \\ 12 \end{pmatrix}; \ = \begin{pmatrix} 4 & 1 \\ 1 & 2 \end{pmatrix}$$

```
library(MASS)
mu = as.matrix(c(5, 12))
Sigma = matrix(c(4, 1, 1, 2), nrow = 2, byrow = T)
sim <- mvrnorm(n = 1000, mu = mu, Sigma = Sigma)
plot(sim[, 1], sim[, 2])
```

Here,

$$\mathbf{A} = \begin{pmatrix} 0.9239 & -0.3827 \\ 0.3827 & 0.9239 \end{pmatrix}$$

Columns of $\mathbf{A}$ are the eigenvectors for the decomposition

Under matrix multiplication ($\mathbf{A}'\ \mathbf{A}$ or $\mathbf{A}'\mathbf{A}$), the off-diagonal elements equal to 0

Multiplying data by this matrix (i.e., projecting the data onto the orthogonal axes); the distribution of the resulting data (i.e., "scores") is

$$N_2(\mathbf{A}'\ , \mathbf{A}'\ \mathbf{A}) = N_2(\mathbf{A}'\ ,\ )$$

Equivalently,

$$\mathbf{y} = \mathbf{A}'\mathbf{x} \sim N\left[\begin{pmatrix} 9.2119 \\ 9.1733 \end{pmatrix}, \begin{pmatrix} 4.4144 & 0 \\ 0 & 1.5859 \end{pmatrix}\right]$$

```
A_matrix = matrix(c(0.9239, -0.3827, 0.3827, 0.9239),
                  nrow = 2,
                  byrow = T)
t(A_matrix) %*% A_matrix
#>          [,1]     [,2]
#> [1,] 1.000051 0.000000
#> [2,] 0.000000 1.000051
```

```
sim1 <-
    mvrnorm(
        n = 1000,
        mu = t(A_matrix) %*% mu,
        Sigma = t(A_matrix) %*% Sigma %*% A_matrix
    )
plot(sim1[, 1], sim1[, 2])
```

No more dependence in the data structure, plot

Notes:

- The i-th eigenvalue is the variance of a linear combination of the elements of $\mathbf{x}$; $var(y_i) = var(\mathbf{a_i'x}) = \lambda_i$
- The values on the transformed set of axes (i.e., the y_i's) are called the scores. These are the orthogonal projections of the data onto the "new principal component axes
- Variances of y_1 are greater than those for any other possible projection

Covariance matrix decomposition and projection onto orthogonal axes = PCA

25.2.1 Population Principal Components

$p \times 1$ vectors $\mathbf{x}_1, \dots, \mathbf{x}_n$ which are iid with $var(\mathbf{x}_i) =$

- The first PC is the linear combination $y_1 = \mathbf{a}_1'\mathbf{x} = a_{11}x_1 + \cdots + a_{1p}x_p$ with $\mathbf{a}_1'\mathbf{a}_1 = 1$ such that $var(y_1)$ is the maximum of all linear combinations of $\mathbf{x}$ which have unit length

- The second PC is the linear combination $y_1 = \mathbf{a}_2'\mathbf{x} = a_{21}x_1 + \cdots + a_{2p}x_p$ with $\mathbf{a}_2'\mathbf{a}_2 = 1$ such that $var(y_1)$ is the maximum of all linear combinations of $\mathbf{x}$ which have unit length and uncorrelated with y_1 (i.e., $cov(\mathbf{a}_1'\mathbf{x}, \mathbf{a}_2'\mathbf{x}) = 0$
- continues for all y_i to y_p

$\mathbf{a}_i$'s are those that make up the matrix $\mathbf{A}$ in the symmetric decomposition $\mathbf{A}'\ \mathbf{A} =$, where $var(y_1) = \lambda_1, \ldots, var(y_p) = \lambda_p$ And the total variance of $\mathbf{x}$ is

$$\begin{aligned} var(x_1) + \cdots + var(x_p) &= tr(\Sigma) = \lambda_1 + \cdots + \lambda_p \\ &= var(y_1) + \cdots + var(y_p) \end{aligned}$$

Data Reduction

To reduce the dimension of data from p (original) to k dimensions without much "loss of information", we can use properties of the population principal components

- Suppose $\approx \sum_{i=1}^{k} \lambda_i \mathbf{a}_i \mathbf{a}_i'$. Even thought the true variance-covariance matrix has rank p , it can be be well approximate by a matrix of rank k (k <p)
- New "traits" are linear combinations of the measured traits. We can attempt to make meaningful interpretation fo the combinations (with orthogonality constraints).
- The proportion of the total variance accounted for by the j-th principal component is

$$\frac{var(y_j)}{\sum_{i=1}^{p} var(y_i)} = \frac{\lambda_j}{\sum_{i=1}^{p} \lambda_i}$$

- The proportion of the total variation accounted for by the first k principal components is $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{p} \lambda_i}$
- Above example , we have $4.4144/(4+2) = .735$ of the total variability can be explained by the first principal component

25.2.2 Sample Principal Components

Since is unknown, we use

$$\mathbf{S} = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})'$$

Let $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \ldots \geq \hat{\lambda}_p \geq 0$ be the eigenvalues of $\mathbf{S}$ and $\hat{\mathbf{a}}_1, \hat{\mathbf{a}}_2, \ldots, \hat{\mathbf{a}}_p$ denote the eigenvectors of $\mathbf{S}$

Then, the i-th sample principal component score (or principal component or score) is

$$\hat{y}_{ij} = \sum_{k=1}^{p} \hat{a}_{ik} x_{kj} = \hat{\mathbf{a}}_i' \mathbf{x}_j$$

Properties of Sample Principal Components

- The estimated variance of $y_i = \hat{\mathbf{a}}_i' \mathbf{x}_j$ is $\hat{\lambda}_i$
- The sample covariance between $\hat{y}_i$ and $\hat{y}_{i'}$ is 0 when $i \neq i'$
- The proportion of the total sample variance accounted for by the i-th sample principal component is $\frac{\hat{\lambda}_i}{\sum_{k=1}^{p} \hat{\lambda}_k}$
- The estimated correlation between the i-th principal component score and the l-th attribute of $\mathbf{x}$ is

$$r_{x_l, \hat{y}_i} = \frac{\hat{a}_{il} \sqrt{\lambda_i}}{\sqrt{s_{ll}}}$$

- The correlation coefficient is typically used to interpret the components (i.e., if this correlation is high then it suggests that the l-th original trait is important in the i-th principle component). According to Johnson et al. (2002), pp.433-434, $r_{x_l, \hat{y}_i}$ only measures the univariate contribution of an individual X to a component Y without taking into account the presence of the other X's. Hence, some prefer $\hat{a}_{il}$ coefficient to interpret the principal component.
- $r_{x_l, \hat{y}_i}; \hat{a}_{il}$ are referred to as "loadings"

To use k principal components, we must calculate the scores for each data vector in the sample

$$\mathbf{y}_j = \begin{pmatrix} y_{1j} \\ y_{2j} \\ \vdots \\ y_{kj} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{a}}_1' \mathbf{x}_j \\ \hat{\mathbf{a}}_2' \mathbf{x}_j \\ \vdots \\ \hat{\mathbf{a}}_k' \mathbf{x}_j \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{a}}_1' \\ \hat{\mathbf{a}}_2' \\ \vdots \\ \hat{\mathbf{a}}_k' \end{pmatrix} \mathbf{x}_j$$

Issues:

- Large sample theory exists for eigenvalues and eigenvectors of sample covariance matrices if inference is necessary. But we do not do inference with PCA, we only use it as exploratory or descriptive analysis.
- PC is not invariant to changes in scale (Exception: if all trait are rescaled by multiplying by the same constant, such as feet to inches).

- PCA based on the correlation matrix $\mathbf{R}$ is different than that based on the covariance matrix
- PCA for the correlation matrix is just rescaling each trait to have unit variance
- Transform $\mathbf{x}$ to $\mathbf{z}$ where $z_{ij} = (x_{ij} - \bar{x}_i)/\sqrt{s_{ii}}$ where the denominator affects the PCA
- After transformation, $cov(\mathbf{z}) = \mathbf{R}$
- PCA on $\mathbf{R}$ is calculated in the same way as that on $\mathbf{S}$ (where $\hat{\lambda}_1 + \cdots + \hat{\lambda}_p = p$)
- The use of $\mathbf{R}, \mathbf{S}$ depends on the purpose of PCA.
  * If the scale of the observations if different, covariance matrix is more preferable. but if they are dramatically different, analysis can still be dominated by the large variance traits.
- How many PCs to use can be guided by
  * Scree Graphs: plot the eigenvalues against their indices. Look for the "elbow" where the steep decline in the graph suddenly flattens out; or big gaps.
  * minimum Percent of total variation (e.g., choose enough components to have 50% or 90%). can be used for interpretations.
  * Kaiser's rule: use only those PC with eigenvalues larger than 1 (applied to PCA on the correlation matrix) - ad hoc
  * Compare to the eigenvalue scree plot of data to the scree plot when the data are randomized.

25.2.3 Application

PCA on the covariance matrix is usually not preferred due to the fact that PCA is not invariant to changes in scale. Hence, PCA on the correlation matrix is more preferred

This also addresses the problem of multicollinearity

The eigvenvectors may differ by a multiplication of -1 for different implementation, but same interpretation.

```
library(tidyverse)
## Read in and check data
stock <- read.table("images/stock.dat")
names(stock) <- c("allied", "dupont", "carbide", "exxon", "texaco")
str(stock)
#> 'data.frame':    100 obs. of  5 variables:
```

```
#>  $ allied : num  0 0.027 0.1228 0.057 0.0637 ...
#>  $ dupont : num  0 -0.04485 0.06077 0.02995 -0.00379 ...
#>  $ carbide: num  0 -0.00303 0.08815 0.06681 -0.03979 ...
#>  $ exxon  : num  0.0395 -0.0145 0.0862 0.0135 -0.0186 ...
#>  $ texaco : num  0 0.0435 0.0781 0.0195 -0.0242 ...

## Covariance matrix of data
cov(stock)
#>               allied       dupont      carbide        exxon       texaco
#> allied  0.0016299269 0.0008166676 0.0008100713 0.0004422405 0.0005139715
#> dupont  0.0008166676 0.0012293759 0.0008276330 0.0003868550 0.0003109431
#> carbide 0.0008100713 0.0008276330 0.0015560763 0.0004872816 0.0004624767
#> exxon   0.0004422405 0.0003868550 0.0004872816 0.0008023323 0.0004084734
#> texaco  0.0005139715 0.0003109431 0.0004624767 0.0004084734 0.0007587370

## Correlation matrix of data
cor(stock)
#>            allied    dupont   carbide     exxon    texaco
#> allied  1.0000000 0.5769244 0.5086555 0.3867206 0.4621781
#> dupont  0.5769244 1.0000000 0.5983841 0.3895191 0.3219534
#> carbide 0.5086555 0.5983841 1.0000000 0.4361014 0.4256266
#> exxon   0.3867206 0.3895191 0.4361014 1.0000000 0.5235293
#> texaco  0.4621781 0.3219534 0.4256266 0.5235293 1.0000000

# cov(scale(stock)) # give the same result

## PCA with covariance
cov_pca <- prcomp(stock)
# uses singular value decomposition for calculation and an N -1 divisor
# alternatively, princomp can do PCA via spectral decomposition,
# but it has worse numerical accuracy

# eigen values
cov_results <- data.frame(eigen_values = cov_pca$sdev ^ 2)
cov_results %>%
    mutate(proportion = eigen_values / sum(eigen_values),
           cumulative = cumsum(proportion))
#>   eigen_values proportion cumulative
#> 1 0.0035953867 0.60159252  0.6015925
#> 2 0.0007921798 0.13255027  0.7341428
#> 3 0.0007364426 0.12322412  0.8573669
#> 4 0.0005086686 0.08511218  0.9424791
#> 5 0.0003437707 0.05752091  1.0000000
# first 2 PCs account for 73% variance in the data
```

```r
# eigen vectors
cov_pca$rotation # prcomp calls rotation
#>               PC1         PC2        PC3         PC4         PC5
#> allied  0.5605914  0.73884565 -0.1260222 -0.28373183  0.20846832
#> dupont  0.4698673 -0.09286987 -0.4675066  0.68793190 -0.28069055
#> carbide 0.5473322 -0.65401929 -0.1140581 -0.50045312  0.09603973
#> exxon   0.2908932 -0.11267353  0.6099196  0.43808002  0.58203935
#> texaco  0.2842017  0.07103332  0.6168831 -0.06227778 -0.72784638
# princomp calls loadings.

# first PC = overall average
# second PC compares Allied to Carbide

## PCA with correlation
#same as scale(stock) %>% prcomp
cor_pca <- prcomp(stock, scale = T)



# eigen values
cor_results <- data.frame(eigen_values = cor_pca$sdev ^ 2)
cor_results %>%
    mutate(proportion = eigen_values / sum(eigen_values),
           cumulative = cumsum(proportion))
#>   eigen_values proportion cumulative
#> 1    2.8564869 0.57129738  0.5712974
#> 2    0.8091185 0.16182370  0.7331211
#> 3    0.5400440 0.10800880  0.8411299
#> 4    0.4513468 0.09026936  0.9313992
#> 5    0.3430038 0.06860076  1.0000000

# first egiven values corresponds to less variance
# than PCA based on the covariance matrix

# eigen vectors
cor_pca$rotation
#>               PC1        PC2        PC3        PC4        PC5
#> allied  0.4635405  0.2408499 -0.6133570  0.3813727 -0.4532876
#> dupont  0.4570764  0.5090997  0.1778996  0.2113068  0.6749814
#> carbide 0.4699804  0.2605774  0.3370355 -0.6640985 -0.3957247
#> exxon   0.4216770 -0.5252647  0.5390181  0.4728036 -0.1794482
#> texaco  0.4213291 -0.5822416 -0.4336029 -0.3812273  0.3874672
# interpretation of PC2 is different from above:
# it is a comparison of Allied, Dupont and Carbid to Exxon and Texaco
```

Covid Example

To reduce collinearity problem in this dataset, we can use principal components as regressors.

```
load('images/MOcovid.RData')
covidpca <- prcomp(ndat[,-1],scale = T,center = T)

covidpca$rotation[,1:2]
#>                                                        PC1         PC2
#> X..Population.in.Rural.Areas                      0.32865838  0.05090955
#> Area..sq..miles.                                  0.12014444 -0.28579183
#> Population.density..sq..miles.                   -0.29670124  0.28312922
#> Literacy.rate                                    -0.12517700 -0.08999542
#> Families                                         -0.25856941  0.16485752
#> Area.of.farm.land..sq..miles.                     0.02101106 -0.31070363
#> Number.of.farms                                  -0.03814582 -0.44809679
#> Average.value.of.all.property.per.farm..dollars. -0.05410709  0.14404306
#> Estimation.of.rurality..                         -0.19040210  0.12089501
#> Male..                                            0.02182394 -0.09568768
#> Number.of.Physcians.per.100.000                  -0.31451606  0.13598026
#> average.age                                       0.29414708  0.35593459
#> X0.4.age.proportion                              -0.11431336 -0.23574057
#> X20.44.age.proportion                            -0.32802128 -0.22718550
#> X65.and.over.age.proportion                       0.30585033  0.32201626
#> prop..White..nonHisp                              0.35627561 -0.14142646
#> prop..Hispanic                                   -0.16655381 -0.15105342
#> prop..Black                                      -0.33333359  0.24405802


# Variability of each principal component: pr.var
pr.var <- covidpca$sdev ^ 2
# Variance explained by each principal component: pve
pve <- pr.var / sum(pr.var)
plot(
    pve,
    xlab = "Principal Component",
    ylab = "Proportion of Variance Explained",
    ylim = c(0, 0.5),
    type = "b"
)
```

```
plot(
    cumsum(pve),
    xlab = "Principal Component",
    ylab = "Cumulative Proportion of Variance Explained",
    ylim = c(0, 1),
    type = "b"
)
```

```r
# the first six principe account for around 80% of the variance.


#using base lm function for PC regression
pcadat <- data.frame(covidpca$x[, 1:6])
pcadat$y <- ndat$Y
pcr.man <- lm(log(y) ~ ., pcadat)
mean(pcr.man$residuals ^ 2)
#> [1] 0.03453371

#comparison to lm w/o prin comps
lm.fit <- lm(log(Y) ~ ., data = ndat)
mean(lm.fit$residuals ^ 2)
#> [1] 0.02335128
```

MSE for the PC-based model is larger than regular regression, because models with a large degree of collinearity can still perform well.

`pcr` function in `pls` can be used for fitting PC regression (it will select the optimal number of components in the model).

25.3 Factor Analysis

Purpose

- Using a few linear combinations of underlying unobservable (latent) traits, we try to describe the covariance relationship among a large number of measured traits
- Similar to PCA, but factor analysis is **model based**

More details can be found on PSU stat or UMN stat

Let $\mathbf{y}$ be the set of p measured variables

$E(\mathbf{y}) =$

$var(\mathbf{y}) =$

We have

$$\begin{aligned}\mathbf{y} - &= \mathbf{L}\mathbf{f} + \epsilon \\ &= \begin{pmatrix} l_{11}f_1 + l_{12}f_2 + \cdots + l_{tm}f_m \\ \vdots \\ l_{p1}f_1 + l_{p2}f_2 + \cdots + l_{pm}f_m \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_p \end{pmatrix}\end{aligned}$$

where

- $\mathbf{y} - \quad =$ the p centered measurements
- $\mathbf{L} = p \times m$ matrix of factor loadings
- $\mathbf{f} =$ unobserved common factors for the population
- $\quad =$ random errors (i.e., variation that is not accounted for by the common factors).

We want m (the number of factors) to be much smaller than p (the number of measured attributes)

Restrictions on the model

- $E(\epsilon) = \mathbf{0}$
- $var(\epsilon) = \Psi_{p\times p} = diag(\psi_1, \dots, \psi_p)$
- $, \mathbf{f}$ are independent
- Additional assumption could be $E(\mathbf{f}) = \mathbf{0}, var(\mathbf{f}) = \mathbf{I}_{m\times m}$ (known as the orthogonal factor model) , which imposes the following covariance structure on $\mathbf{y}$

$$\begin{aligned} var(\mathbf{y}) = \quad &= var(\mathbf{Lf} + \quad) \\ &= var(\mathbf{Lf}) + var(\epsilon) \\ &= \mathbf{L}var(\mathbf{f})\mathbf{L}' + \\ &= \mathbf{LIL}' + \\ &= \mathbf{LL}' + \end{aligned}$$

Since is diagonal, the off-diagonal elements of $\mathbf{LL}'$ are σ_{ij}, the co variances in , which means $cov(y_i, y_j) = \sum_{k=1}^{m} l_{ik} l_{jk}$ and the covariance of $\mathbf{y}$ is completely determined by the m factors ($m << p$)

$var(y_i) = \sum_{k=1}^{m} l_{ik}^2 + \psi_i$ where ψ_i is the **specific variance** and the summation term is the i-th **communality** (i.e., portion of the variance of the i-th variable contributed by the m common factors ($h_i^2 = \sum_{k=1}^{m} l_{ik}^2$)

The factor model is only uniquely determined up to an orthogonal transformation of the factors.

Let $\mathbf{T}_{m\times m}$ be an orthogonal matrix $\mathbf{TT}' = \mathbf{T}'\mathbf{T} = \mathbf{I}$ then

$$\begin{aligned} \mathbf{y} - \quad &= \mathbf{Lf} + \epsilon \\ &= \mathbf{LTT}'\mathbf{f} + \epsilon \\ &= \mathbf{L}^*(\mathbf{T}'\mathbf{f}) + \epsilon \quad \text{where } \mathbf{L}^* = \mathbf{LT} \end{aligned}$$

and

$$
\begin{aligned}
&= \mathbf{L}\mathbf{L}' + \\
&= \mathbf{L}\mathbf{T}\mathbf{T}'\mathbf{L} + \\
&= (\mathbf{L}^*)(\mathbf{L}^*)' +
\end{aligned}
$$

Hence, any orthogonal transformation of the factors is an equally good description of the correlations among the observed traits.

Let $\mathbf{y} = \mathbf{C}\mathbf{x}$, where $\mathbf{C}$ is any diagonal matrix, then $\mathbf{L}_y = \mathbf{C}\mathbf{L}_x$ and $_y = \mathbf{C}\ _x\mathbf{C}$

Hence, we can see that factor analysis is also invariant to changes in scale

25.3.1 Methods of Estimation

To estimate $\mathbf{L}$

1. Principal Component Method
2. Principal Factor Method
3. 25.3.1.3

25.3.1.1 Principal Component Method

Spectral decomposition

$$
\begin{aligned}
&= \lambda_1 \mathbf{a}_1 \mathbf{a}_1' + \cdots + \lambda_p \mathbf{a}_p \mathbf{a}_p' \\
&= \mathbf{A}\ \mathbf{A}' \\
&= \sum_{k=1}^{m} \lambda + k \mathbf{a}_k \mathbf{a}_k' + \sum_{k=m+1}^{p} \lambda_k \mathbf{a}_k \mathbf{a}_k' \\
&= \sum_{k=1}^{m} l_k l_k' + \sum_{k=m+1}^{p} \lambda_k \mathbf{a}_k \mathbf{a}_k'
\end{aligned}
$$

where $l_k = \mathbf{a}_k \sqrt{\lambda_k}$ and the second term is not diagonal in general.

Assume

$$
\psi_i = \sigma_{ii} - \sum_{k=1}^{m} l_{ik}^2 = \sigma_{ii} - \sum_{k=1}^{m} \lambda_i a_{ik}^2
$$

then

$$
\approx \mathbf{L}\mathbf{L}' +
$$

To estimate $\mathbf{L}$ and Ψ , we use the expected eigenvalues and eigenvectors from $\mathbf{S}$ or $\mathbf{R}$

- The estimated factor loadings don't change as the number of actors increases
- The diagonal elements of $\hat{\mathbf{L}}\hat{\mathbf{L}}' + \hat{}$ are equal to the diagonal elements of $\mathbf{S}$ and $\mathbf{R}$, but the covariances may not be exactly reproduced
- We select m so that the off-diagonal elements close to the values in $\mathbf{S}$ (or to make the off-diagonal elements of $\mathbf{S} - \hat{\mathbf{L}}\hat{\mathbf{L}}' + \hat{}$ small)

25.3.1.2 Principal Factor Method

Consider modeling the correlation matrix, $\mathbf{R} = \mathbf{LL}' + \quad$. Then

$$\mathbf{LL}' = \mathbf{R} - \quad = \begin{pmatrix} h_1^2 & r_{12} & \dots & r_{1p} \\ r_{21} & h_2^2 & \dots & r_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ r_{p1} & r_{p2} & \dots & h_p^2 \end{pmatrix}$$

where $h_i^2 = 1 - \psi_i$ (the communality)

Suppose that initial estimates are available for the communalities, $(h_1^*)^2, (h_2^*)^2, \dots, (h_p^*)^2$, then we can regress each trait on all the others, and then use the r^2 as h^2

The estimate of $\mathbf{R} - \quad$ at step k is

$$(\mathbf{R} - \quad)_k = \begin{pmatrix} (h_1^*)^2 & r_{12} & \dots & r_{1p} \\ r_{21} & (h_2^*)^2 & \dots & r_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ r_{p1} & r_{p2} & \dots & (h_p^*)^2 \end{pmatrix} = \mathbf{L}_k^*(\mathbf{L}_k^*)'$$

where

$$\mathbf{L}_k^* = (\sqrt{\hat{\lambda}_1^* \hat{\mathbf{a}}_1^*, \dots \hat{\lambda}_m^* \hat{\mathbf{a}}_m^*})$$

and

$$\hat{\psi}_{i,k}^* = 1 - \sum_{j=1}^{m} \hat{\lambda}_i^* (\hat{a}_{ij}^*)^2$$

we used the spectral decomposition on the estimated matrix $(\mathbf{R} - \quad)$ to calculate the $\hat{\lambda}_i^* s$ and the $\hat{\mathbf{a}}_i^* s$

After updating the values of $(\hat{h}_i^*)^2 = 1 - \hat{\psi}_{i,k}^*$ we will use them to form a new $\mathbf{L}_{k+1}^*$ via another spectral decomposition. Repeat the process

Notes:

- The matrix $(\mathbf{R} - \ )_k$ is not necessarily positive definite
- The principal component method is similar to principal factor if one considers the initial communalities are $h^2 = 1$
- if m is too large, some communalities may become larger than 1, causing the iterations to terminate. To combat, we can
  - fix any communality that is greater than 1 at 1 and then continues.
  - continue iterations regardless of the size of the communalities. However, results can be outside fo the parameter space.

25.3.1.3 Maximum Likelihood Method

Since we need the likelihood function, we make the additional (critical) assumption that

- $\mathbf{y}_j \sim N(\ ,\ )$ for $j = 1, .., n$
- $\mathbf{f} \sim N(\mathbf{0}, \mathbf{I})$
- $\epsilon_j \sim N(\mathbf{0},\ )$

and restriction

- $\mathbf{L}'\ ^{-1}\mathbf{L} =$ where is a diagonal matrix. (since the factor loading matrix is not unique, we need this restriction).

Notes:

- Finding MLE can be computationally expensive
- we typically use other methods for exploratory data analysis
- Likelihood ratio tests could be used for testing hypotheses in this framework (i.e., Confirmatory Factor Analysis)

25.3.2 Factor Rotation

$\mathbf{T}_{m\times m}$ is an orthogonal matrix that has the property that

$$\hat{\mathbf{L}}\hat{\mathbf{L}}' + \hat{\ } = \hat{\mathbf{L}}^*(\hat{\mathbf{L}}^*)' + \hat{\ }$$

where $\mathbf{L}^* = \mathbf{LT}$

This means that estimated specific variances and communalities are not altered by the orthogonal transformation.

Since there are an infinite number of choices for $\mathbf{T}$, some selection criterion is necessary

For example, we can find the orthogonal transformation that maximizes the objective function

$$\sum_{j=1}^{m}[\frac{1}{p}\sum_{i=1}^{p}(\frac{l_{ij}^{*2}}{h_i})^2 - \{\frac{\gamma}{p}\sum_{i=1}^{p}(\frac{l_{ij}^{*2}}{h_i})^2\}^2]$$

where $\frac{l_{ij}^{*2}}{h_i}$ are "scaled loadings", which gives variables with small communalities more influence.

Different choices of γ in the objective function correspond to different orthogonal rotation found in the literature;

1. Varimax $\gamma = 1$ (rotate the factors so that each of the p variables should have a high loading on only one factor, but this is not always possible).
2. Quartimax $\gamma = 0$
3. Equimax $\gamma = m/2$
4. Parsimax $\gamma = \frac{p(m-1)}{p+m-2}$
5. Promax: non-orthogonal or olique transformations
6. Harris-Kaiser (HK): non-orthogonal or oblique transformations

25.3.3 Estimation of Factor Scores

Recall

$$(\mathbf{y}_j - \ ) = \mathbf{L}_{p\times m}\mathbf{f}_j + \epsilon_j$$

If the factor model is correct then

$$var(\epsilon_j) = \ = diag(\psi_1, \dots, \psi_p)$$

Thus we could consider using weighted least squares to estimate $\mathbf{f}_j$, the vector of factor scores for the j-th sampled unit by

$$\begin{aligned}\hat{\mathbf{f}} &= (\mathbf{L}'\ ^{-1}\mathbf{L})^{-1}\mathbf{L}'\ ^{-1}(\mathbf{y}_j - \ ) \\ &\approx (\mathbf{L}'\ ^{-1}\mathbf{L})^{-1}\mathbf{L}'\ ^{-1}(\mathbf{y}_j - \bar{\mathbf{y}})\end{aligned}$$

25.3.3.1 The Regression Method

Alternatively, we can use the regression method to estimate the factor scores

Consider the joint distribution of $(\mathbf{y}_j - \ )$ and $\mathbf{f}_j$ assuming multivariate normality, as in the maximum likelihood approach. then,

$$\begin{pmatrix} \mathbf{y}_j - \\ \mathbf{f}_j \end{pmatrix} \sim N_{p+m}\left(\begin{bmatrix} \mathbf{L}\mathbf{L}' + & \mathbf{L} \\ \mathbf{L}' & \mathbf{I}_{m\times m} \end{bmatrix}\right)$$

when the m factor model is correct

Hence,

$$E(\mathbf{f}_j|\mathbf{y}_j - \ ) = \mathbf{L}'(\mathbf{L}\mathbf{L}' + \ )^{-1}(\mathbf{y}_j - \ )$$

notice that $\mathbf{L}'(\mathbf{L}\mathbf{L}' + \ )^{-1}$ is an $m \times p$ matrix of regression coefficients

Then, we use the estimated conditional mean vector to estimate the factor scores

$$\hat{\mathbf{f}}_j = \hat{\mathbf{L}}'(\hat{\mathbf{L}}\hat{\mathbf{L}}' + \hat{} \ )^{-1}(\mathbf{y}_j - \bar{\mathbf{y}})$$

Alternatively, we could reduce the effect of possible incorrect determination fo the number of factors m by using $\mathbf{S}$ as a substitute for $\hat{\mathbf{L}}\hat{\mathbf{L}}' + \hat{} \ \ \text{then}$

$$\hat{\mathbf{f}}_j = \hat{\mathbf{L}}'\mathbf{S}^{-1}(\mathbf{y}_j - \bar{\mathbf{y}})$$

where $j = 1, \ldots, n$

25.3.4 Model Diagnostic

- Plots
- Check for outliers (recall that $\mathbf{f}_j \sim iidN(\mathbf{0}, \mathbf{I}_{m \times m})$)
- Check for multivariate normality assumption
- Use univariate tests for normality to check the factor scores
- **Confirmatory Factor Analysis**: formal testing of hypotheses about loadings, use MLE and full/reduced model testing paradigm and measures of model fit

25.3.5 Application

In the `psych` package,

- h2 = the communalities
- u2 = the uniqueness
- com = the complexity

```
library(psych)
library(tidyverse)
## Load the data from the psych package
data(Harman.5)
Harman.5
#>         population schooling employment professional housevalue
```

```
#> Tract1        5700      12.8       2500          270      25000
#> Tract2        1000      10.9        600           10      10000
#> Tract3        3400       8.8       1000           10       9000
#> Tract4        3800      13.6       1700          140      25000
#> Tract5        4000      12.8       1600          140      25000
#> Tract6        8200       8.3       2600           60      12000
#> Tract7        1200      11.4        400           10      16000
#> Tract8        9100      11.5       3300           60      14000
#> Tract9        9900      12.5       3400          180      18000
#> Tract10       9600      13.7       3600          390      25000
#> Tract11       9600       9.6       3300           80      12000
#> Tract12       9400      11.4       4000          100      13000

# Correlation matrix
cor_mat <- cor(Harman.5)
cor_mat
#>              population  schooling employment professional housevalue
#> population   1.00000000 0.00975059  0.9724483    0.4388708 0.02241157
#> schooling    0.00975059 1.00000000  0.1542838    0.6914082 0.86307009
#> employment   0.97244826 0.15428378  1.0000000    0.5147184 0.12192599
#> professional 0.43887083 0.69140824  0.5147184    1.0000000 0.77765425
#> housevalue   0.02241157 0.86307009  0.1219260    0.7776543 1.00000000

## Principal Component Method with Correlation
cor_pca <- prcomp(Harman.5, scale = T)
# eigen values
cor_results <- data.frame(eigen_values = cor_pca$sdev ^ 2)

cor_results <- cor_results %>%
    mutate(
        proportion = eigen_values / sum(eigen_values),
        cumulative = cumsum(proportion),
        number = row_number()
    )
cor_results
#>   eigen_values  proportion cumulative number
#> 1   2.87331359 0.574662719  0.5746627      1
#> 2   1.79666009 0.359332019  0.9339947      2
#> 3   0.21483689 0.042967377  0.9769621      3
#> 4   0.09993405 0.019986811  0.9969489      4
#> 5   0.01525537 0.003051075  1.0000000      5

# Scree plot of Eigenvalues
scree_gg <- ggplot(cor_results, aes(x = number, y = eigen_values)) +
    geom_line(alpha = 0.5) +
```

```
    geom_text(aes(label = number)) +
    scale_x_continuous(name = "Number") +
    scale_y_continuous(name = "Eigenvalue") +
    theme_bw()
scree_gg
```

```
screeplot(cor_pca, type = 'lines')
```

```r
## Keep 2 factors based on scree plot and eigenvalues
factor_pca <- principal(Harman.5, nfactors = 2, rotate = "none")
factor_pca
#> Principal Components Analysis
#> Call: principal(r = Harman.5, nfactors = 2, rotate = "none")
#> Standardized loadings (pattern matrix) based upon correlation matrix
#>               PC1   PC2   h2    u2 com
#> population   0.58  0.81 0.99 0.012 1.8
#> schooling    0.77 -0.54 0.89 0.115 1.8
#> employment   0.67  0.73 0.98 0.021 2.0
#> professional 0.93 -0.10 0.88 0.120 1.0
#> housevalue   0.79 -0.56 0.94 0.062 1.8
#>
#>                        PC1  PC2
#> SS loadings           2.87 1.80
#> Proportion Var        0.57 0.36
#> Cumulative Var        0.57 0.93
#> Proportion Explained  0.62 0.38
#> Cumulative Proportion 0.62 1.00
#>
#> Mean item complexity =  1.7
#> Test of the hypothesis that 2 components are sufficient.
#>
#> The root mean square of the residuals (RMSR) is  0.03
#>  with the empirical chi square  0.29  with prob <  0.59
#>
#> Fit based upon off diagonal values = 1

# factor 1 = overall socioeconomic health
# factor 2 = contrast of the population and employment against school and house value


## Ssquared multiple correlation (SMC) prior, no rotation
factor_pca_smc <- fa(
    Harman.5,
    nfactors = 2,
    fm = "pa",
    rotate = "none",
    SMC = TRUE
)
factor_pca_smc
#> Factor Analysis using method =  pa
#> Call: fa(r = Harman.5, nfactors = 2, rotate = "none", SMC = TRUE, fm = "pa")
#> Standardized loadings (pattern matrix) based upon correlation matrix
#>               PA1   PA2   h2      u2 com
```

```
#> population   0.62  0.78 1.00 -0.0027 1.9
#> schooling    0.70 -0.53 0.77  0.2277 1.9
#> employment   0.70  0.68 0.96  0.0413 2.0
#> professional 0.88 -0.15 0.80  0.2017 1.1
#> housevalue   0.78 -0.60 0.96  0.0361 1.9
#>
#>                        PA1  PA2
#> SS loadings           2.76 1.74
#> Proportion Var        0.55 0.35
#> Cumulative Var        0.55 0.90
#> Proportion Explained  0.61 0.39
#> Cumulative Proportion 0.61 1.00
#>
#> Mean item complexity =  1.7
#> Test of the hypothesis that 2 factors are sufficient.
#>
#> df null model =  10  with the objective function =  6.38 with Chi Square =  54.25
#> df of  the model are 1  and the objective function was  0.34
#>
#> The root mean square of the residuals (RMSR) is  0.01
#> The df corrected root mean square of the residuals is  0.03
#>
#> The harmonic n.obs is  12 with the empirical chi square  0.02  with prob <  0.88
#> The total n.obs was  12  with Likelihood Chi Square =  2.44  with prob <  0.12
#>
#> Tucker Lewis Index of factoring reliability =  0.596
#> RMSEA index =  0.336  and the 90 % confidence intervals are  0 0.967
#> BIC =  -0.04
#> Fit based upon off diagonal values = 1

## SMC prior, Promax rotation
factor_pca_smc_pro <- fa(
    Harman.5,
    nfactors = 2,
    fm = "pa",
    rotate = "Promax",
    SMC = TRUE
)
factor_pca_smc_pro
#> Factor Analysis using method =  pa
#> Call: fa(r = Harman.5, nfactors = 2, rotate = "Promax", SMC = TRUE,
#>     fm = "pa")
#> Standardized loadings (pattern matrix) based upon correlation matrix
#>                PA1   PA2   h2      u2 com
#> population   -0.11  1.02 1.00 -0.0027 1.0
```

```
#> schooling     0.90 -0.11 0.77  0.2277 1.0
#> employment    0.02  0.97 0.96  0.0413 1.0
#> professional  0.75  0.33 0.80  0.2017 1.4
#> housevalue    1.01 -0.14 0.96  0.0361 1.0
#>
#>                        PA1  PA2
#> SS loadings           2.38 2.11
#> Proportion Var        0.48 0.42
#> Cumulative Var        0.48 0.90
#> Proportion Explained  0.53 0.47
#> Cumulative Proportion 0.53 1.00
#>
#>  With factor correlations of
#>      PA1  PA2
#> PA1 1.00 0.25
#> PA2 0.25 1.00
#>
#> Mean item complexity =  1.1
#> Test of the hypothesis that 2 factors are sufficient.
#>
#> df null model =  10  with the objective function =  6.38 with Chi Square =  54.25
#> df of  the model are 1  and the objective function was  0.34
#>
#> The root mean square of the residuals (RMSR) is  0.01
#> The df corrected root mean square of the residuals is  0.03
#>
#> The harmonic n.obs is  12 with the empirical chi square  0.02  with prob <  0.88
#> The total n.obs was  12  with Likelihood Chi Square =  2.44  with prob <  0.12
#>
#> Tucker Lewis Index of factoring reliability =  0.596
#> RMSEA index =  0.336  and the 90 % confidence intervals are  0 0.967
#> BIC =  -0.04
#> Fit based upon off diagonal values = 1

## SMC prior, varimax rotation
factor_pca_smc_var <- fa(
    Harman.5,
    nfactors = 2,
    fm = "pa",
    rotate = "varimax",
    SMC = TRUE
)
## Make a data frame of the loadings for ggplot2
factors_df <-
    bind_rows(
```

```
        data.frame(
            y = rownames(factor_pca_smc$loadings),
            unclass(factor_pca_smc$loadings)
        ),
        data.frame(
            y = rownames(factor_pca_smc_pro$loadings),
            unclass(factor_pca_smc_pro$loadings)
        ),
        data.frame(
            y = rownames(factor_pca_smc_var$loadings),
            unclass(factor_pca_smc_var$loadings)
        ),
        .id = "Rotation"
    )
flag_gg <- ggplot(factors_df) +
    geom_vline(aes(xintercept = 0)) +
    geom_hline(aes(yintercept = 0)) +
    geom_point(aes(
        x = PA2,
        y = PA1,
        col = y,
        shape = y
    ), size = 2) +
    scale_x_continuous(name = "Factor 2", limits = c(-1.1, 1.1)) +
    scale_y_continuous(name = "Factor1", limits = c(-1.1, 1.1)) +
    facet_wrap("Rotation", labeller = labeller(Rotation = c(
        "1" = "Original", "2" = "Promax", "3" = "Varimax"
    ))) +
    coord_fixed(ratio = 1) # make aspect ratio of each facet 1

flag_gg
```

```
# promax and varimax did a good job to assign trait to a particular factor

factor_mle_1 <- fa(
    Harman.5,
    nfactors = 1,
    fm = "mle",
    rotate = "none",
    SMC = TRUE
)
factor_mle_1
#> Factor Analysis using method =  ml
#> Call: fa(r = Harman.5, nfactors = 1, rotate = "none", SMC = TRUE, fm = "mle")
#> Standardized loadings (pattern matrix) based upon correlation matrix
#>               ML1    h2     u2 com
#> population   0.97 0.950 0.0503   1
#> schooling    0.14 0.021 0.9791   1
#> employment   1.00 0.995 0.0049   1
#> professional 0.51 0.261 0.7388   1
#> housevalue   0.12 0.014 0.9864   1
#>
#>                 ML1
#> SS loadings    2.24
#> Proportion Var 0.45
#>
#> Mean item complexity =  1
#> Test of the hypothesis that 1 factor is sufficient.
```

```
#>
#> df null model =  10  with the objective function =  6.38 with Chi Square =  54.25
#> df of  the model are 5  and the objective function was  3.14
#>
#> The root mean square of the residuals (RMSR) is  0.41
#> The df corrected root mean square of the residuals is  0.57
#>
#> The harmonic n.obs is  12 with the empirical chi square  39.41  with prob <  2e-07
#> The total n.obs was  12  with Likelihood Chi Square =  24.56  with prob <  0.00017
#>
#> Tucker Lewis Index of factoring reliability =  0.022
#> RMSEA index =  0.564  and the 90 % confidence intervals are  0.374 0.841
#> BIC =  12.14
#> Fit based upon off diagonal values = 0.5
#> Measures of factor score adequacy
#>                                                   ML1
#> Correlation of (regression) scores with factors   1.00
#> Multiple R square of scores with factors          1.00
#> Minimum correlation of possible factor scores     0.99

factor_mle_2 <- fa(
    Harman.5,
    nfactors = 2,
    fm = "mle",
    rotate = "none",
    SMC = TRUE
)
factor_mle_2
#> Factor Analysis using method =  ml
#> Call: fa(r = Harman.5, nfactors = 2, rotate = "none", SMC = TRUE, fm = "mle")
#> Standardized loadings (pattern matrix) based upon correlation matrix
#>                ML2  ML1   h2    u2 com
#> population   -0.03 1.00 1.00 0.005 1.0
#> schooling     0.90 0.04 0.81 0.193 1.0
#> employment    0.09 0.98 0.96 0.036 1.0
#> professional  0.78 0.46 0.81 0.185 1.6
#> housevalue    0.96 0.05 0.93 0.074 1.0
#>
#>                        ML2  ML1
#> SS loadings           2.34 2.16
#> Proportion Var        0.47 0.43
#> Cumulative Var        0.47 0.90
#> Proportion Explained  0.52 0.48
#> Cumulative Proportion 0.52 1.00
#>
```

```
#> Mean item complexity =  1.1
#> Test of the hypothesis that 2 factors are sufficient.
#>
#> df null model =  10  with the objective function =  6.38 with Chi Square =  54.25
#> df of  the model are 1  and the objective function was  0.31
#>
#> The root mean square of the residuals (RMSR) is  0.01
#> The df corrected root mean square of the residuals is  0.05
#>
#> The harmonic n.obs is  12 with the empirical chi square  0.05  with prob <  0.82
#> The total n.obs was  12  with Likelihood Chi Square =  2.22  with prob <  0.14
#>
#> Tucker Lewis Index of factoring reliability =  0.658
#> RMSEA index =  0.307  and the 90 % confidence intervals are  0 0.945
#> BIC =  -0.26
#> Fit based upon off diagonal values = 1
#> Measures of factor score adequacy
#>                                                   ML2  ML1
#> Correlation of (regression) scores with factors   0.98 1.00
#> Multiple R square of scores with factors          0.95 1.00
#> Minimum correlation of possible factor scores     0.91 0.99

factor_mle_3 <- fa(
    Harman.5,
    nfactors = 3,
    fm = "mle",
    rotate = "none",
    SMC = TRUE
)
factor_mle_3
#> Factor Analysis using method =  ml
#> Call: fa(r = Harman.5, nfactors = 3, rotate = "none", SMC = TRUE, fm = "mle")
#> Standardized loadings (pattern matrix) based upon correlation matrix
#>                ML2  ML1   ML3   h2     u2 com
#> population   -0.12 0.98 -0.11 0.98 0.0162 1.1
#> schooling     0.89 0.15  0.29 0.90 0.0991 1.3
#> employment    0.00 1.00  0.04 0.99 0.0052 1.0
#> professional  0.72 0.52 -0.10 0.80 0.1971 1.9
#> housevalue    0.97 0.13 -0.09 0.97 0.0285 1.1
#>
#>                        ML2  ML1  ML3
#> SS loadings           2.28 2.26 0.11
#> Proportion Var        0.46 0.45 0.02
#> Cumulative Var        0.46 0.91 0.93
#> Proportion Explained  0.49 0.49 0.02
```

```
#> Cumulative Proportion 0.49 0.98 1.00
#>
#> Mean item complexity =  1.2
#> Test of the hypothesis that 3 factors are sufficient.
#>
#> df null model =  10  with the objective function =  6.38 with Chi Square =  54.25
#> df of  the model are -2  and the objective function was  0
#>
#> The root mean square of the residuals (RMSR) is  0
#> The df corrected root mean square of the residuals is  NA
#>
#> The harmonic n.obs is  12 with the empirical chi square  0  with prob <  NA
#> The total n.obs was  12  with Likelihood Chi Square =  0  with prob <  NA
#>
#> Tucker Lewis Index of factoring reliability =  1.318
#> Fit based upon off diagonal values = 1
#> Measures of factor score adequacy
#>                                                   ML2  ML1  ML3
#> Correlation of (regression) scores with factors   0.99 1.00 0.82
#> Multiple R square of scores with factors          0.98 1.00 0.68
#> Minimum correlation of possible factor scores     0.96 0.99 0.36
```

The output info for the null hypothesis of no common factors is in the statement "The degrees of freedom for the null model .."

The output info for the null hypothesis that number of factors is sufficient is in the statement "The total number of observations was …"

One factor is not enough, two is sufficient, and not enough data for 3 factors (df of -2 and NA for p-value). Hence, we should use 2-factor model.

25.4 Discriminant Analysis

Suppose we have two or more different populations from which observations could come from. Discriminant analysis seeks to determine which of the possible population an observation comes from while making as few mistakes as possible

- This is an alternative to logistic approaches with the following advantages:
  - when there is clear separation between classes, the parameter estimates for the logic regression model can be **surprisingly** unstable, while discriminant approaches do not suffer
  - If X is normal in each of the classes and the sample size is small, then discriminant approaches can be more accurate

Notation

Similar to MANOVA, let $\mathbf{y}_{j1}, \mathbf{y}_{j2}, \dots, \mathbf{y}_{in_j} \sim iid f_j(\mathbf{y})$ for $j = 1, \dots, h$

Let $f_j(\mathbf{y})$ be the density function for population j . Note that each vector $\mathbf{y}$ contain measurements on all p traits

1. Assume that each observation is from one of h possible populations.
2. We want to form a discriminant rule that will allocate an observation $\mathbf{y}$ to population j when $\mathbf{y}$ is in fact from this population

25.4.1 Known Populations

The maximum likelihood discriminant rule for assigning an observation $\mathbf{y}$ to one of the h populations allocates $\mathbf{y}$ to the population that gives the largest likelihood to $\mathbf{y}$

Consider the likelihood for a single observation $\mathbf{y}$, which has the form $f_j(\mathbf{y})$ where j is the true population.

Since j is unknown, to make the likelihood as large as possible, we should choose the value j which causes $f_j(\mathbf{y})$ to be as large as possible

Consider a simple univariate example. Suppose we have data from one of two binomial populations.

- The first population has $n = 10$ trials with success probability $p = .5$
- The second population has $n = 10$ trials with success probability $p = .7$
- to which population would we assign an observation of $y = 7$
- Note:
  - $f(y = 7|n = 10, p = .5) = .117$
  - $f(y = 7|n = 10, p = .7) = .267$ where $f(.)$ is the binomial likelihood.
  - Hence, we choose the second population

Another example

We have 2 populations, where

- First population: $N(\mu_1, \sigma_1^2)$
- Second population: $N(\mu_2, \sigma_2^2)$

The likelihood for a single observation is

$$f_j(y) = (2\pi\sigma_j^2)^{-1/2} \exp\{-\frac{1}{2}(\frac{y - \mu_j}{\sigma_j})^2\}$$

Consider a likelihood ratio rule

$$\begin{aligned}\Lambda &= \frac{\text{likelihood of y from pop 1}}{\text{likelihood of y from pop 2}} \\ &= \frac{f_1(y)}{f_2(y)} \\ &= \frac{\sigma_2}{\sigma_1}\exp\{-\frac{1}{2}[(\frac{y-\mu_1}{\sigma_1})^2 - (\frac{y-\mu_2}{\sigma_2})^2]\}\end{aligned}$$

Hence, we classify into

- pop 1 if $\Lambda > 1$
- pop 2 if $\Lambda < 1$
- for ties, flip a coin

Another way to think:

we classify into population 1 if the "standardized distance" of y from μ_1 is less than the "standardized distance" of y from μ_2 which is referred to as a **quadratic discriminant rule**.

(Significant simplification occurs in th special case where $\sigma_1 = \sigma_2 = \sigma^2$)

Thus, we classify into population 1 if

$$(y-\mu_2)^2 > (y-\mu_1)^2$$

or

$$|y-\mu_2| > |y-\mu_1|$$

and

$$-2\log(\Lambda) = -2y\frac{(\mu_1-\mu_2)}{\sigma^2} + \frac{(\mu_1^2-\mu_2^2)}{\sigma^2} = \beta y + \alpha$$

Thus, we classify into population 1 if this is less than 0.

Discriminant classification rule is linear in y in this case.

25.4.1.1 Multivariate Expansion

Suppose that there are 2 populations

- $N_p(\ _1,\ _1)$
- $N_p(\ _2,\ _2)$

$$
\begin{aligned}
-2\log(\frac{f_1(\mathbf{x})}{f_2(\mathbf{x})}) &= \log|\ _1| + (\mathbf{x} - \ _1)'\ _1^{-1}(\mathbf{x} - \ _1) \\
&\quad - [\log|\ _2| + (\mathbf{x} - \ _2)'\ _2^{-1}(\mathbf{x} - \ _2)]
\end{aligned}
$$

Again, we classify into population 1 if this is less than 0, otherwise, population 2. And like the univariate case with non-equal variances, this is a quadratic discriminant rule.

And if the covariance matrices are equal: $_1 = \ _2 = \ _1$ classify into population 1 if

$$
(\ _1 - \ _2)'\ ^{-1}\mathbf{x} - \frac{1}{2}(\ _1 - \ _2)'\ ^{-1}(\ _1 - \ _2) \geq 0
$$

This linear discriminant rule is also referred to as **Fisher's linear discriminant function**

By **assuming the covariance matrices are equal, we assume that the shape and orientation fo the two populations must be the same (which can be a strong restriction)**

In other words, for each variable, it can have different mean but the same variance.

- Note: LDA Bayes decision boundary is linear. Hence, quadratic decision boundary might lead to better classification. Moreover, the assumption of same variance/covariance matrix across all classes for Gaussian densities imposes the linear rule, if we allow the predictors in each class to follow MVN distribution with class-specific mean vectors and variance/covariance matrices, then it is **Quadratic Discriminant Analysis.** But then, you will have more parameters to estimate (which gives more flexibility than LDA) at the cost of more variance (bias -variance tradeoff).

When $_1, \ _2, \ $ are known, the probability of misclassification can be determined:

$$
\begin{aligned}
P(2|1) &= P(\text{calssify into pop 2| x is from pop 1}) \\
&= P((\ _1 - \ _2)'\ ^{-1}\mathbf{x} \leq \frac{1}{2}(\ _1 - \ _2)'\ ^{-1}(\ _1 - \ _2)|\mathbf{x} \sim N(\mu_1, \ ) \\
&= \Phi(-\frac{1}{2}\delta)
\end{aligned}
$$

where

- $\delta^2 = (\ _1 - \ _2)'\ ^{-1}(\ _1 - \ _2)$
- Φ is the standard normal CDF

Suppose there are h possible populations, which are distributed as $N_p(\ _p,\ )$. Then, the maximum likelihood (linear) discriminant rule allocates $\mathbf{y}$ to population j where j minimizes the squared Mahalanobis distance

$$(\mathbf{y} - \ _j)' \ ^{-1}(\mathbf{y} - \ _j)$$

25.4.1.2 Bayes Discriminant Rules

If we know that population j has prior probabilities π_j (assume $\pi_j > 0$) we can form the Bayes discriminant rule.

This rule allocates an observation $\mathbf{y}$ to the population for which $\pi_j f_j(\mathbf{y})$ is maximized.

Note:

- **Maximum likelihood discriminant rule** is a special case of the **Bayes discriminant rule**, where it sets all the $\pi_j = 1/h$

Optimal Properties of Bayes Discriminant Rules

- let p_{ii} be the probability of correctly assigning an observation from population i
- then one rule (with probabilities p_{ii}) is as good as another rule (with probabilities p'_{ii}) if $p_{ii} \geq p'_{ii}$ for all $i = 1, \ldots, h$
- The first rule is better than the alternative if $p_{ii} > p'_{ii}$ for at least one i.
- A rule for which there is no better alternative is called admissible
- Bayes Discriminant Rules are admissible
- If we utilized prior probabilities, then we can form the posterior probability of a correct allocation, $\sum_{i=1}^{h} \pi_i p_{ii}$
- Bayes Discriminant Rules have the largest possible posterior probability of correct allocation with respect to the prior
- These properties show that **Bayes Discriminant rule is our best approach**.

Unequal Cost

- We want to consider the cost misallocation
  - Define c_{ij} to be the cost associated with allocation a member of population j to population i.
- Assume that
  - $c_{ij} > 0$ for all $i \neq j$
  - $c_{ij} = 0$ if $i = j$

- We could determine the expected amount of loss for an observation allocated to population i as $\sum_j c_{ij} p_{ij}$ where the $p_{ij}s$ are the probabilities of allocating an observation from population j into population i
- We want to minimize the amount of loss expected for our rule. Using a Bayes Discrimination, allocate $\mathbf{y}$ to the population j which minimizes $\sum_{k \neq j} c_{ij} \pi_k f_k(\mathbf{y})$
- We could assign equal probabilities to each group and get a maximum likelihood type rule. here, we would allocate $\mathbf{y}$ to population j which minimizes $\sum_{k \neq j} c_{jk} f_k(\mathbf{y})$

Example:

Two binomial populations, each of size 10, with probabilities $p_1 = .5$ and $p_2 = .7$

And the probability of being in the first population is .9

However, suppose the cost of inappropriately allocating into the first population is 1 and the cost of incorrectly allocating into the second population is 5.

In this case, we pick population 1 over population 2

In general, we consider two regions, R_1 and R_2 associated with population 1 and 2:

$$R_1 : \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} \geq \frac{c_{12}\pi_2}{c_{21}\pi_1}$$

$$R_2 : \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} < \frac{c_{12}\pi_2}{c_{21}\pi_1}$$

where c_{12} is the cost of assigning a member of population 2 to population 1.

25.4.1.3 Discrimination Under Estimation

Suppose we know the form of the distributions for populations of interests, but we still have to estimate the parameters.

Example:

we know the distributions are multivariate normal, but we have to estimate the means and variances

The maximum likelihood discriminant rule allocates an observation $\mathbf{y}$ to population j when j maximizes the function

$$f_j(\mathbf{y}|\hat{\theta})$$

where $\hat{\theta}$ are the maximum likelihood estimates of the unknown parameters

For instance, we have 2 multivariate normal populations with distinct means, but common variance covariance matrix

MLEs for $_1$ and $_2$ are $\bar{\mathbf{y}}_1$ and $\bar{\mathbf{y}}_2$and common is $\mathbf{S}$.

Thus, an estimated discriminant rule could be formed by substituting these sample values for the population values

25.4.1.4 Native Bayes

- The challenge with classification using Bayes' is that we don't know the (true) densities, $f_k, k = 1, \dots, K$, while LDA and QDA make **strong multivariate normality assumptions** to deal with this.
- Naive Bayes makes only one assumption: **within the k-th class, the p predictors are independent (i.e,, for** $k = 1, \dots, K$

$$f_k(x) = f_{k1}(x_1) \times f_{k2}(x_2) \times \cdots \times f_{kp}(x_p)$$

where f_{kj} is the density function of the j-th predictor among observation in the k-th class.

This assumption allows the use of joint distribution without the need to account for dependence between observations. However, this (native) assumption can be unrealistic, but still works well in cases where the number of sample (n) is not large relative to the number of features (p).

With this assumption, we have

$$P(Y = k|X = x) = \frac{\pi_k \times f_{k1}(x_1) \times \cdots \times f_{kp}(x_p)}{\sum_{l=1}^{K} \pi_l \times f_{l1}(x_1) \times \dots f_{lp}(x_p)}$$

we only need to estimate the one-dimensional density function f_{kj} with either of these approaches:

- When X_j is quantitative, assume it has a univariate normal distribution (with independence): $X_j|Y = k \sim N(\mu_{jk}, \sigma_{jk}^2)$ which is more restrictive than QDA because it assumes predictors are independent (e.g., a diagonal covariance matrix)
- When X_j is quantitative, use a kernel density estimator Kernel Methods ; which is a smoothed histogram
- When X_j is qualitative, we count the promotion of training observations for the j-th predictor corresponding to each class.

25.4.1.5 Comparison of Classification Methods

Assuming we have K classes and K is the baseline from (James , Witten, Hastie, and Tibshirani book)

Comparing the log odds relative to the K class

25.4.1.5.1 Logistic Regression

$$\log(\frac{P(Y=k|X=x)}{P(Y=K|X=x)}) = \beta_{k0} + \sum_{j=1}^{p} \beta_{kj} x_j$$

25.4.1.5.2 LDA

$$\log(\frac{P(Y=k|X=x)}{P(Y=K|X=x)} = a_k + \sum_{j=1}^{p} b_{kj} x_j$$

where a_k and b_{kj} are functions of $\pi_k, \pi_K, \mu_k, \mu_K$,

Similar to logistic regression, LDA assumes the log odds is linear in x

Even though they look like having the same form, the parameters in logistic regression are estimated by MLE, where as LDA linear parameters are specified by the prior and normal distributions

We expect LDA to outperform logistic regression when the normality assumption (approximately) holds, and logistic regression to perform better when it does not

25.4.1.5.3 QDA

$$\log(\frac{P(Y=k|X=x}{P(Y=K|X=x}) = a_k + \sum_{j=1}^{p} b_{kj} x_j + \sum_{j=1}^{p} \sum_{l=1}^{p} c_{kjl} x_j x_l$$

where a_k, b_{kj}, c_{kjl} are functions $\pi_k, \pi_K, \mu_k, \mu_K, \ _k, \ _K$

25.4.1.5.4 Naive Bayes

$$\log(\frac{P(Y=k|X=x)}{P(Y=K|X=x}) = a_k + \sum_{j=1}^{p} g_{kj}(x_j)$$

where $a_k = \log(\pi_k/\pi_K)$ and $g_{kj}(x_j) = \log(\frac{f_{kj}(x_j)}{f_{Kj}(x_j)})$ which is the form of generalized additive model

25.4.1.5.5 Summary

- LDA is a special case of QDA
- LDA is robust when it comes to high dimensions
- Any classifier with a linear decision boundary is a special case of naive Bayes with $g_{kj}(x_j) = b_{kj}x_j$, which means LDA is a special case of naive Bayes. LDA assumes that the features are normally distributed with a common within-class covariance matrix, and naive Bayes assumes independence of the features.
- Naive bayes is also a special case of LDA with restricted to a diagonal matrix with diagonals, σ^2 (another notation $diag(\ )$) assuming $f_{kj}(x_j) = N(\mu_{kj}, \sigma_j^2)$
- QDA and naive Bayes are not special case of each other. In principal,e naive Bayes can produce a more flexible fit by the choice of $g_{kj}(x_j)$, but it's restricted to only purely additive fit, but QDA includes multiplicative terms of the form $c_{kjl}x_jx_l$
- None of these methods uniformly dominates the others: the choice of method depends on the true distribution of the predictors in each of the K classes, n and p (i.e., related to the bias-variance tradeoff).

Compare to the non-parametric method (KNN)

- KNN would outperform both LDA and logistic regression when the decision boundary is highly nonlinear, but can't say which predictors are most important, and requires many observations
- KNN is also limited in high-dimensions due to the curse of dimensionality
- Since QDA is a special type of nonlinear decision boundary (quadratic), it can be considered as a compromise between the linear methods and KNN classification. QDA can have fewer training observations than KNN but not as flexible.

From simulation:

True decision boundary	Best performance
Linear	LDA + Logistic regression
Moderately nonlinear	QDA + Naive Bayes
Highly nonlinear (many training, p is not large)	KNN

- like linear regression, we can also introduce flexibility by including transformed features $\sqrt{X}, X^2, X^3$

25.4.2 Probabilities of Misclassification

When the distribution are exactly known, we can determine the misclassification probabilities exactly. however, when we need to estimate the population parameters, we have to estimate the probability of misclassification

- Naive method
  - Plugging the parameters estimates into the form for the misclassification probabilities results to derive at the estimates of the misclassification probability.
  - But this will tend to be optimistic when the number of samples in one or more populations is small.
- Resubstitution method
  - Use the proportion of the samples from population i that would be allocated to another population as an estimate of the misclassification probability
  - But also optimistic when the number of samples is small
- Jack-knife estimates:
  - The above two methods use observation to estimate both parameters and also misclassification probabilities based upon the discriminant rule
  - Alternatively, we determine the discriminant rule based upon all of the data except the k-th observation from the j-th population
  - then, determine if the k-th observation would be misclassified under this rule
  - perform this process for all n_j observation in population j . An estimate fo the misclassification probability would be the fraction of n_j observations which were misclassified
  - repeat the process for other $i \neq j$ populations
  - This method is more reliable than the others, but also computationally intensive
- Cross-Validation

Summary

Consider the group-specific densities $f_j(\mathbf{x})$ for multivariate vector $\mathbf{x}$.

Assume equal misclassifications costs, the Bayes classification probability of $\mathbf{x}$ belonging to the j-th population is

$$p(j|\mathbf{x}) = \frac{\pi_j f_j(\mathbf{x})}{\sum_{k=1}^{h} \pi_k f_k(\mathbf{x})}$$

$j = 1, \dots, h$

where there are h possible groups.

We then classify into the group for which this probability of membership is largest

Alternatively, we can write this in terms of a **generalized squared distance** formation

$$D_j^2(\mathbf{x}) = d_j^2(\mathbf{x}) + g_1(j) + g_2(j)$$

where

- $d_j^2(\mathbf{x}) = (\mathbf{x} - \ _j)'\mathbf{V}_j^{-1}(\mathbf{x} - \ _j)$ is the squared Mahalanobis distance from $\mathbf{x}$ to the centroid of group j, and
  - $\mathbf{V}_j = \mathbf{S}_j$ if the within group covariance matrices are not equal
  - $\mathbf{V}_j = \mathbf{S}_p$ if a pooled covariance estimate is appropriate

and

$$g_1(j) = \begin{cases} \ln|\mathbf{S}_j| & \text{within group covariances are not equal} \\ 0 & \text{pooled covariance} \end{cases}$$

$$g_2(j) = \begin{cases} -2\ln\pi_j & \text{prior probabilities are not equal} \\ 0 & \text{prior probabilities are equal} \end{cases}$$

then, the posterior probability of belonging to group j is

$$p(j|\mathbf{x}) = \frac{\exp(-.5D_j^2(\mathbf{x}))}{\sum_{k=1}^{h} \exp(-.5D_k^2(\mathbf{x}))}$$

where $j = 1, \dots, h$

and $\mathbf{x}$ is classified into group j if $p(j|\mathbf{x})$ is largest for $j = 1, \dots, h$ (or, $D_j^2(\mathbf{x})$ is smallest).

25.4.2.1 Assessing Classification Performance

For binary classification, confusion matrix

	Predicted class			
		- or Null	+ or Null	Total
True Class	- or Null	True Neg (TN)	False Pos (FP)	N
	+ or Null	False Neg (FN)	True Pos (TP)	P
	Total	N*	P*	

and table 4.6 from (James et al., 2013)

Name	Definition	Synonyms
False Pos rate	FP/N	Type I error, 1 0 Specificity
True Pos. rate	TP/P	1 - Type II error, power, sensitivity, recall
Pos Pred. value	TP/P*	Precision, 1 - false discovery promotion
Neg. Pred. value	TN/N*	

ROC curve (receiver Operating Characteristics) is a graphical comparison between **sensitivity** (true positive) and **specificity** (= 1 - false positive)

y-axis = true positive rate

x-axis = false positive rate

as we change the threshold rate for classifying an observation as from 0 to 1

AUC (area under the ROC) ideally would equal to 1, a bad classifier would have AUC = 0.5 (pure chance)

25.4.3 Unknown Populations/ Nonparametric Discrimination

When your multivariate data are not Gaussian, or known distributional form at all, we can use the following methods

25.4.3.1 Kernel Methods

We approximate $f_j(\mathbf{x})$ by a kernel density estimate

$$\hat{f}_j(\mathbf{x}) = \frac{1}{n_j} \sum_{i=1}^{n_j} K_j(\mathbf{x} - \mathbf{x}_i)$$

where

- $K_j(.)$ is a kernel function satisfying $\int K_j(\mathbf{z}) d\mathbf{z} = 1$

- $\mathbf{x}_i$, $i = 1, \dots, n_j$ is a random sample from the j-th population.

Thus, after finding $\hat{f}_j(\mathbf{x})$ for each of the h populations, the posterior probability of group membership is

$$p(j|\mathbf{x}) = \frac{\pi_j \hat{f}_j(\mathbf{x})}{\sum_{k-1}^{h} \pi_k \hat{f}_k(\mathbf{x})}$$

where $j = 1, \dots, h$

There are different choices for the kernel function:

- Uniform
- Normal
- Epanechnikov
- Biweight
- Triweight

We these kernels, we have to pick the "radius" (or variance, width, window width, bandwidth) of the kernel, which is a smoothing parameter (the larger the radius, the more smooth the kernel estimate of the density).

To select the smoothness parameter, we can use the following method

If we believe the populations were close to multivariate normal, then

$$R = (\frac{4/(2p+1)}{n_j})^{1/(p+1}$$

But since we do not know for sure, we might choose several different values and select one that vies the best out of sample or cross-validation discrimination.

Moreover, you also have to decide whether to use different kernel smoothness for different populations, which is similar to the individual and pooled covariances in the classical methodology.

25.4.3.2 Nearest Neighbor Methods

The nearest neighbor (also known as k-nearest neighbor) method performs the classification of a new observation vector based on the group membership of its nearest neighbors. In practice, we find

$$d_{ij}^2(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x}, \mathbf{x}_i) V_j^{-1} (\mathbf{x}, \mathbf{x}_i)$$

which is the distance between the vector $\mathbf{x}$ and the i-th observation in group j

We consider different choices for $\mathbf{V}_j$

For example,

$$\begin{aligned}
\mathbf{V}_j &= \mathbf{S}_p \\
\mathbf{V}_j &= \mathbf{S}_j \\
\mathbf{V}_j &= \mathbf{I} \\
\mathbf{V}_j &= diag(\mathbf{S}_p)
\end{aligned}$$

We find the k observations that are closest to $\mathbf{x}$ (where users pick k). Then we classify into the most common population, weighted by the prior.

25.4.3.3 Modern Discriminant Methods

Note:

Logistic regression (with or without random effects) is a flexible model-based procedure for classification between two populations.

The extension of logistic regression to the multi-group setting is polychotomous logistic regression (or, mulinomial regression).

The machine learning and pattern recognition are growing with strong focus on nonlinear discriminant analysis methods such as:

- radial basis function networks
- support vector machines
- multiplayer perceptrons (neural networks)

The general framework

$$g_j(\mathbf{x}) = \sum_{l=1}^{m} w_{jl}\phi_l(\mathbf{x};\ _l) + w_{j0}$$

where

- $j = 1, \dots, h$
- m nonlinear basis functions ϕ_l, each of which has n_m parameters given by $\theta_l = \{\theta_{lk} : k = 1, \dots, n_m\}$

We assign $\mathbf{x}$ to the j-th population if $g_j(\mathbf{x})$ is the maximum for all $j = 1, \dots, h$

Development usually focuses on the choice and estimation of the basis functions, ϕ_l and the estimation of the weights w_{jl}

More details can be found (Webb et al., 2011)

25.4.4 Application

```
library(class)
library(klaR)
library(MASS)
library(tidyverse)

## Read in the data
crops <- read.table("images/crops.txt")
names(crops) <- c("crop", "y1", "y2", "y3", "y4")
str(crops)
#> 'data.frame':    36 obs. of  5 variables:
#>  $ crop: chr  "Corn" "Corn" "Corn" "Corn" ...
#>  $ y1  : int  16 15 16 18 15 15 12 20 24 21 ...
#>  $ y2  : int  27 23 27 20 15 32 15 23 24 25 ...
#>  $ y3  : int  31 30 27 25 31 32 16 23 25 23 ...
#>  $ y4  : int  33 30 26 23 32 15 73 25 32 24 ...


## Read in test data
crops_test <- read.table("images/crops_test.txt")
names(crops_test) <- c("crop", "y1", "y2", "y3", "y4")
str(crops_test)
#> 'data.frame':    5 obs. of  5 variables:
#>  $ crop: chr  "Corn" "Soybeans" "Cotton" "Sugarbeets" ...
#>  $ y1  : int  16 21 29 54 32
#>  $ y2  : int  27 25 24 23 32
#>  $ y3  : int  31 23 26 21 62
#>  $ y4  : int  33 24 28 54 16
```

25.4.4.1 LDA

Default prior is proportional to sample size and `lda` and `qda` do not fit a constant or intercept term

```
## Linear discriminant analysis
lda_mod <- lda(crop ~ y1 + y2 + y3 + y4,
               data = crops)
lda_mod
#> Call:
#> lda(crop ~ y1 + y2 + y3 + y4, data = crops)
#>
#> Prior probabilities of groups:
#>     Clover       Corn     Cotton   Soybeans Sugarbeets
#>  0.3055556  0.1944444  0.1666667  0.1666667  0.1666667
#>
```

```
#> Group means:
#>                  y1       y2       y3       y4
#> Clover     46.36364 32.63636 34.18182 36.63636
#> Corn       15.28571 22.71429 27.42857 33.14286
#> Cotton     34.50000 32.66667 35.00000 39.16667
#> Soybeans   21.00000 27.00000 23.50000 29.66667
#> Sugarbeets 31.00000 32.16667 20.00000 40.50000
#>
#> Coefficients of linear discriminants:
#>              LD1          LD2         LD3          LD4
#> y1 -6.147360e-02  0.009215431 -0.02987075 -0.014680566
#> y2 -2.548964e-02  0.042838972  0.04631489  0.054842132
#> y3  1.642126e-02 -0.079471595  0.01971222  0.008938745
#> y4  5.143616e-05 -0.013917423  0.05381787 -0.025717667
#>
#> Proportion of trace:
#>    LD1    LD2    LD3    LD4
#> 0.7364 0.1985 0.0576 0.0075

## Look at accuracy on the training data
lda_fitted <- predict(lda_mod,newdata = crops)
# Contingency table
lda_table <- table(truth = crops$crop, fitted = lda_fitted$class)
lda_table
#>             fitted
#> truth        Clover Corn Cotton Soybeans Sugarbeets
#>   Clover          6    0      3        0          2
#>   Corn            0    6      0        1          0
#>   Cotton          3    0      1        2          0
#>   Soybeans        0    1      1        3          1
#>   Sugarbeets      1    1      0        2          2
# accuracy of 0.5 is just random (not good)

## Posterior probabilities of membership
crops_post <- cbind.data.frame(crops,
                               crop_pred = lda_fitted$class,
                               lda_fitted$posterior)
crops_post <- crops_post %>%
    mutate(missed = crop != crop_pred)
head(crops_post)
#>   crop y1 y2 y3 y4 crop_pred     Clover      Corn    Cotton  Soybeans
#> 1 Corn 16 27 31 33      Corn 0.08935164 0.4054296 0.1763189 0.2391845
#> 2 Corn 15 23 30 30      Corn 0.07690181 0.4558027 0.1420920 0.2530101
#> 3 Corn 16 27 27 26      Corn 0.09817815 0.3422454 0.1365315 0.3073105
#> 4 Corn 18 20 25 23      Corn 0.10521511 0.3633673 0.1078076 0.3281477
```

```
#> 5 Corn 15 15 31 32      Corn 0.05879921 0.5753907 0.1173332 0.2086696
#> 6 Corn 15 32 32 15  Soybeans 0.09723648 0.3278382 0.1318370 0.3419924
#>   Sugarbeets missed
#> 1 0.08971545  FALSE
#> 2 0.07219340  FALSE
#> 3 0.11573442  FALSE
#> 4 0.09546233  FALSE
#> 5 0.03980738  FALSE
#> 6 0.10109590   TRUE
# posterior shows that posterior of corn membership is much higher than the prior

## LOOCV
# leave-one-out cross validation for linear discriminant analysis
# cannot run the predict function using the object with CV = TRUE
# because it returns the within sample predictions
lda_cv <- lda(crop ~ y1 + y2 + y3 + y4,
              data = crops, CV = TRUE)
# Contingency table
lda_table_cv <- table(truth = crops$crop, fitted = lda_cv$class)
lda_table_cv
#>             fitted
#> truth        Clover Corn Cotton Soybeans Sugarbeets
#>   Clover          4    3      1        0          3
#>   Corn            0    4      1        2          0
#>   Cotton          3    0      0        2          1
#>   Soybeans        0    1      1        3          1
#>   Sugarbeets      2    1      0        2          1

## Predict the test data
lda_pred <- predict(lda_mod, newdata = crops_test)

## Make a contingency table with truth and most likely class
table(truth=crops_test$crop, predict=lda_pred$class)
#>             predict
#> truth        Clover Corn Cotton Soybeans Sugarbeets
#>   Clover          0    0      1        0          0
#>   Corn            0    1      0        0          0
#>   Cotton          0    0      0        1          0
#>   Soybeans        0    0      0        1          0
#>   Sugarbeets      1    0      0        0          0
```

LDA didn't do well on both within sample and out-of-sample data.

25.4.4.2 QDA

```
## Quadratic discriminant analysis
qda_mod <- qda(crop ~ y1 + y2 + y3 + y4,
               data = crops)

## Look at accuracy on the training data
qda_fitted <- predict(qda_mod, newdata = crops)
# Contingency table
qda_table <- table(truth = crops$crop, fitted = qda_fitted$class)
qda_table
#>             fitted
#> truth        Clover Corn Cotton Soybeans Sugarbeets
#>   Clover          9    0      0        0          2
#>   Corn            0    7      0        0          0
#>   Cotton          0    0      6        0          0
#>   Soybeans        0    0      0        6          0
#>   Sugarbeets      0    0      1        1          4

## LOOCV
qda_cv <- qda(crop ~ y1 + y2 + y3 + y4,
              data = crops, CV = TRUE)
# Contingency table
qda_table_cv <- table(truth = crops$crop, fitted = qda_cv$class)
qda_table_cv
#>             fitted
#> truth        Clover Corn Cotton Soybeans Sugarbeets
#>   Clover          9    0      0        0          2
#>   Corn            3    2      0        0          2
#>   Cotton          3    0      2        0          1
#>   Soybeans        3    0      0        2          1
#>   Sugarbeets      3    0      1        1          1

## Predict the test data
qda_pred <- predict(qda_mod, newdata = crops_test)
## Make a contingency table with truth and most likely class
table(truth = crops_test$crop, predict = qda_pred$class)
#>             predict
#> truth        Clover Corn Cotton Soybeans Sugarbeets
#>   Clover          1    0      0        0          0
#>   Corn            0    1      0        0          0
#>   Cotton          0    0      1        0          0
#>   Soybeans        0    0      0        1          0
#>   Sugarbeets      0    0      0        0          1
```

25.4.4.3 KNN

knn uses design matrices of the features.

```
## Design matrices
X_train <- crops %>%
    dplyr::select(-crop)
X_test <- crops_test %>%
    dplyr::select(-crop)
Y_train <- crops$crop
Y_test <- crops_test$crop

## Nearest neighbors with 2 neighbors
knn_2 <- knn(X_train, X_train, Y_train, k = 2)
table(truth = Y_train, fitted = knn_2)
#>             fitted
#> truth        Clover Corn Cotton Soybeans Sugarbeets
#>   Clover          7    0      3        0          1
#>   Corn            0    6      0        1          0
#>   Cotton          0    0      5        0          1
#>   Soybeans        0    0      0        4          2
#>   Sugarbeets      0    0      2        0          4

## Accuracy
mean(Y_train==knn_2)
#> [1] 0.7222222

## Performance on test data
knn_2_test <- knn(X_train, X_test, Y_train, k = 2)
table(truth = Y_test, predict = knn_2_test)
#>             predict
#> truth        Clover Corn Cotton Soybeans Sugarbeets
#>   Clover          1    0      0        0          0
#>   Corn            0    1      0        0          0
#>   Cotton          0    0      0        0          1
#>   Soybeans        0    0      0        1          0
#>   Sugarbeets      0    0      0        0          1

## Accuracy
mean(Y_test==knn_2_test)
#> [1] 0.8

## Nearest neighbors with 3 neighbors
knn_3 <- knn(X_train, X_train, Y_train, k = 3)
table(truth = Y_train, fitted = knn_3)
#>             fitted
```

```
#> truth        Clover Corn Cotton Soybeans Sugarbeets
#>   Clover          8    0      1        2          0
#>   Corn            0    5      1        1          0
#>   Cotton          1    0      4        1          0
#>   Soybeans        0    0      1        3          2
#>   Sugarbeets      0    0      1        1          4

## Accuracy
mean(Y_train==knn_3)
#> [1] 0.6666667

## Performance on test data
knn_3_test <- knn(X_train, X_test, Y_train, k = 3)
table(truth = Y_test, predict = knn_3_test)
#>             predict
#> truth        Clover Corn Cotton Soybeans Sugarbeets
#>   Clover          1    0      0        0          0
#>   Corn            0    1      0        0          0
#>   Cotton          0    0      1        0          0
#>   Soybeans        0    0      0        1          0
#>   Sugarbeets      0    0      0        0          1

## Accuracy
mean(Y_test==knn_3_test)
#> [1] 1
```

25.4.4.4 Stepwise

Stepwise discriminant analysis using the `stepclass` in function in the `klaR` package.

```
step <- stepclass(
    crop ~ y1 + y2 + y3 + y4,
    data = crops,
    method = "qda",
    improvement = 0.15
)
#> correctness rate: 0.43333;  in: "y1";  variables (1): y1
#>
#>  hr.elapsed min.elapsed sec.elapsed
#>        0.00        0.00        0.22

step$process
#>    step var varname result.pm
#> 0 start   0      -- 0.0000000
#> 1    in   1      y1 0.4333333
```

```
step$performance.measure
#> [1] "correctness rate"
```

Iris Data

```
library(dplyr)
data('iris')
set.seed(1)
samp <-
    sample.int(nrow(iris), size = floor(0.70 * nrow(iris)), replace = F)

train.iris <- iris[samp,] %>% mutate_if(is.numeric,scale)
test.iris <- iris[-samp,] %>% mutate_if(is.numeric,scale)

library(ggplot2)
iris.model <- lda(Species ~ ., data = train.iris)
#pred
pred.lda <- predict(iris.model, test.iris)
table(truth = test.iris$Species, prediction = pred.lda$class)
#>             prediction
#> truth        setosa versicolor virginica
#>   setosa         15          0         0
#>   versicolor      0         17         0
#>   virginica       0          0        13

plot(iris.model)
```

```
iris.model.qda <- qda(Species~.,data=train.iris)
#pred
pred.qda <- predict(iris.model.qda,test.iris)
table(truth=test.iris$Species,prediction=pred.qda$class)
#>              prediction
#> truth         setosa versicolor virginica
#>   setosa         15          0         0
#>   versicolor      0         16         1
#>   virginica       0          0        13
```

25.4.4.5 PCA with Discriminant Analysis

we can use both PCA for dimension reduction in discriminant analysis

```
zeros <- as.matrix(read.table("images/mnist0_train_b.txt"))
nines <- as.matrix(read.table("images/mnist9_train_b.txt"))
train <- rbind(zeros[1:1000, ], nines[1:1000, ])
train <- train / 255 #divide by 255 per notes (so ranges from 0 to 1)
train <- t(train) #each column is an observation
image(matrix(train[, 1], nrow = 28), main = 'Example image, unrotated')
```

```
test <- rbind(zeros[2501:3000, ], nines[2501:3000, ])
test <- test / 255
test <- t(test)
y.train <- c(rep(0, 1000), rep(9, 1000))
```

```
y.test <- c(rep(0, 500), rep(9, 500))


library(MASS)
pc <- prcomp(t(train))
train.large <- data.frame(cbind(y.train, pc$x[, 1:10]))
large <- lda(y.train ~ ., data = train.large)
#the test data set needs to be constucted w/ the same 10 princomps
test.large <- data.frame(cbind(y.test, predict(pc, t(test))[, 1:10]))
pred.lda <- predict(large, test.large)
table(truth = test.large$y.test, prediction = pred.lda$class)
#>      prediction
#> truth   0   9
#>     0 491   9
#>     9   5 495

large.qda <- qda(y.train~.,data=train.large)
#prediction
pred.qda <- predict(large.qda,test.large)
table(truth=test.large$y.test,prediction=pred.qda$class)
#>      prediction
#> truth   0   9
#>     0 493   7
#>     9   3 497
```

B.
QUASI-EXPERIMENTAL DESIGN

Chapter 26

Quasi-experimental

In most cases, it means that you have pre- and post-intervention data.

Great resources for causal inference include Causal Inference Mixtape and Recent Advances in Micro, especially if you like to read about the history of causal inference as a field as well (codes for Stata, R, and Python).

Libraries in R:

- Econometrics
- Causal Inference

Identification strategy for any quasi-experiment (No ways to prove or formal statistical test, but you can provide plausible argument and evidence)

1. Where the exogenous variation comes from (by argument and institutional knowledge)
2. Exclusion restriction: Evidence that the variation in the exogenous shock and the outcome is due to no other factors
    1. The stable unit treatment value assumption (SUTVA) states that the treatment of unit i affect only the outcome of unit i (i.e., no spillover to the control groups)

All quasi-experimental methods involve a tradeoff between power and support for the exogeneity assumption (i.e., discard variation in the data that is not exogenous).

Consequently, we don't usually look at R^2 (Ebbes et al., 2011). And it can even be misleading to use R^2 as the basis for model comparison.

Clustering should be based on the design, not the expectations of correlation (Abadie et al., 2023). With a **small sample**, you should use the **wild bootstrap procedure** (Cameron et al., 2008) to correct for the downward bias (see (Cai et al., 2022)for additional assumptions).

Typical robustness check: recommended by (Goldfarb et al., 2022)

- Different controls: show models with and without controls. Typically, we want to see the change in the estimate of interest. See (Altonji et al., 2005) for a formal assessment based on Rosenbaum bounds (i.e., changes in the estimate and threat of Omitted variables on the estimate). For specific applications in marketing, see (Manchanda et al., 2015) (Shin et al., 2012)
- Different functional forms
- Different window of time (in longitudinal setting)
- Different dependent variables (those that are related) or different measures of the dependent variables
- Different control group size (matched vs. un-matched samples)
- Placebo tests: see each placebo test for each setting below.

Showing the mechanism:

- Mediation analysis
- Moderation analysis
  - Estimate the model separately (for different groups)
  - Assess whether the three-way interaction between the source of variation (e.g., under DID, cross-sectional and time series) and group membership is significant.

External Validity:

- Assess how representative your sample is
- Explain the limitation of the design.
- Use quasi-experimental results in conjunction with structural models: see (Anderson et al., 2015; Einav et al., 2010; Chung et al., 2014)

Limitation

1. What is your identifying assumptions or identification strategy
2. What are threats to the validity of your assumptions?
3. What you do to address it? And maybe how future research can do to address it.

26.1 Natural Experiments

Reusing the same natural experiments for research, particularly when employing identical methods to determine the treatment effect in a given setting, can pose problems for hypothesis testing.

Simulations show that when $N_{\text{Outcome}} >> N_{\text{True effect}}$, more than 50% of statistically significant findings may be false positives (Heath et al., 2023, p.2331).

Solutions:

- Bonferroni correction
- Romano and Wolf (2005) and Romano and Wolf (2016) correction: recommended
- Benjamini and Yekutieli (2001) correction
- Alternatively, refer to the rules of thumb from Table AI (Heath et al., 2023, p.2356).

When applying multiple testing corrections, we can either use (but they will give similar results anyway (Heath et al., 2023, p.2335)):

1. **Chronological Sequencing**: Outcomes are ordered by the date they were first reported, with multiple testing corrections applied in this sequence. This method progressively raises the statistical significance threshold as more outcomes are reviewed over time.
2. **Best Foot Forward Policy**: Outcomes are ordered from most to least likely to be rejected based on experimental data. Used primarily in clinical trials, this approach gives priority to intended treatment effects, which are subjected to less stringent statistical requirements. New outcomes are added to the sequence as they are linked to the primary treatment effect.

```r
# Romano-Wolf correction
library(fixest)
library(wildrwolf)

head(iris)
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#> 1          5.1         3.5          1.4         0.2  setosa
#> 2          4.9         3.0          1.4         0.2  setosa
#> 3          4.7         3.2          1.3         0.2  setosa
#> 4          4.6         3.1          1.5         0.2  setosa
#> 5          5.0         3.6          1.4         0.2  setosa
#> 6          5.4         3.9          1.7         0.4  setosa

fit1 <- feols(Sepal.Width ~ Sepal.Length , data = iris)
fit2 <- feols(Petal.Length ~ Sepal.Length, data = iris)
fit3 <- feols(Petal.Width ~ Sepal.Length, data = iris)

res <- rwolf(
  models = list(fit1, fit2, fit3),
  param = "Sepal.Length",
  B = 500
```

```
)
#>
  |
  |                                                                      |   0%
  |
  |========================                                              |  33%
  |
  |================================================                      |  67%
  |
  |======================================================================| 100%

res
#>   model   Estimate Std. Error   t value     Pr(>|t|) RW Pr(>|t|)
#> 1     1 -0.0618848 0.04296699 -1.440287    0.1518983 0.123752495
#> 2     2   1.858433 0.08585565  21.64602 1.038667e-47 0.001996008
#> 3     3  0.7529176 0.04353017  17.29645 2.325498e-37 0.001996008
```

For all other tests, one can use `multtest::mt.rawp2adjp` which includes:

- Bonferroni
- Holm (1979)
- Šidák (1967)
- Hochberg (1988)
- Benjamini and Hochberg (1995)
- Benjamini and Yekutieli (2001)
- Adaptive Benjamini and Hochberg (2000)
- Two-stage Benjamini et al. (2006)

Permutation adjusted p-values for simple multiple testing procedures

```
# BiocManager::install("multtest")
library(multtest)

procs <-
    c("Bonferroni",
      "Holm",
      "Hochberg",
      "SidakSS",
      "SidakSD",
      "BH",
      "BY",
      "ABH",
      "TSBH")

mt.rawp2adjp(
    # p-values
    runif(10),
```

```
    procs) |> causalverse::nice_tab()
#>    adjp.rawp adjp.Bonferroni adjp.Holm adjp.Hochberg adjp.SidakSS adjp.SidakSD
#> 1       0.12               1         1          0.75         0.72         0.72
#> 2       0.22               1         1          0.75         0.92         0.89
#> 3       0.24               1         1          0.75         0.94         0.89
#> 4       0.29               1         1          0.75         0.97         0.91
#> 5       0.36               1         1          0.75         0.99         0.93
#> 6       0.38               1         1          0.75         0.99         0.93
#> 7       0.44               1         1          0.75         1.00         0.93
#> 8       0.59               1         1          0.75         1.00         0.93
#> 9       0.65               1         1          0.75         1.00         0.93
#> 10      0.75               1         1          0.75         1.00         0.93
#>    adjp.BH adjp.BY adjp.ABH adjp.TSBH_0.05 index h0.ABH h0.TSBH
#> 1     0.63       1     0.63           0.63     2     10      10
#> 2     0.63       1     0.63           0.63     6     10      10
#> 3     0.63       1     0.63           0.63     8     10      10
#> 4     0.63       1     0.63           0.63     3     10      10
#> 5     0.63       1     0.63           0.63    10     10      10
#> 6     0.63       1     0.63           0.63     1     10      10
#> 7     0.63       1     0.63           0.63     7     10      10
#> 8     0.72       1     0.72           0.72     9     10      10
#> 9     0.72       1     0.72           0.72     5     10      10
#> 10    0.75       1     0.75           0.75     4     10      10
```

Chapter 27

Regression Discontinuity

- A regression discontinuity occurs when there is a discrete change (jump) in treatment likelihood in the distribution of a continuous (or roughly continuous) variable (i.e., **running/forcing/assignment variable**).
  - Running variable can also be time, but the argument for time to be continuous is hard to argue because usually we do not see increment of time (e.g., quarterly or annual data). Unless we have minute or hour data, then we might be able to argue for it.
- Review paper (Imbens and Lemieux, 2008; Lee and Lemieux, 2010)
- Other readings:
  - https://ies.ed.gov/ncee/wwc/Docs/ReferenceResources/wwc_rd.pdf
  - https://ies.ed.gov/ncee/wwc/Docs/ReferenceResources/wwc_rdd_standards_122315.pdf
- (Thistlethwaite and Campbell, 1960): first paper to use RD in the context of merit awards on future academic outcomes.
- RD is a localized experiment at the cutoff point
  - Hence, we always have to qualify (perfunctory) our statement in research articles that "our research might not generalize to beyond the bandwidth."
- In reality, RD and experimental (from random assignment) estimates are very similar ((Chaplin et al., 2018); Mathematica). But still, it's hard to prove empirically for every context (there might be future study that finds a huge difference between local estimate - causal - and overall estimate - random assignment.

- Threats: only valid near threshold: inference at threshold is valid on average. Interestingly, random experiment showed the validity already.
- Tradeoff between efficiency and bias
- Regression discontinuity is under the framework of Instrumental Variable (structural IV) argued by (Angrist and Lavy, 1999) and a special case of the Matching Methods (matching at one point) argued by (Heckman et al., 1999).
- The hard part is to find a setting that can apply, but once you find one, it's easy to apply
- We can also have multiple cutoff lines. However, for each cutoff line, there can only be one breakup point
- RD can have multiple coinciding effects (i.e., joint distribution or bundled treatment), then RD effect in this case would be the joint effect.
- As the running variable becomes more discrete your framework should be Interrupted Time Series, but for more granular levels you can use RD. When you have infinite data (or substantially large) the two frameworks are identical. RD is always better than Interrupted Time Series
- Multiple alternative model specifications that produce consistent results are more reliable (parametric - linear regression with polynomials terms, and non-parametric - local linear regression). This is according to (Lee and Lemieux, 2010), one straightforward method to ease the linearity assumption is by incorporating polynomial functions of the forcing variable. The choice of polynomial terms can be determined based on the data.
  - . According to (Gelman and Imbens, 2019), accounting for global high-order polynomials presents three issues: (1) imprecise estimates due to noise, (2) sensitivity to the polynomial's degree, and (3) inadequate coverage of confidence intervals. To address this, researchers should instead employ estimators that rely on local linear or quadratic polynomials or other smooth functions.
- RD should be viewed more as a description of a data generating process, rather than a method or approach (similar to a randomized experiment)
- RD is close to
  - other quasi-experimental methods in the sense that it's based on the discontinuity at a threshold
  - randomized experiments in the sense that it's local randomization.

There are several types of Regression Discontinuity:

1. Sharp RD: Change in treatment probability at the cutoff point is 1

- Kink design: Instead of a discontinuity in the level of running variable, we have a discontinuity in the slope of the function (while the function/level can remain continuous) (Nielsen et al., 2010). See (Böckerman et al., 2018) for application, and (Card et al., 2015) for theory.

2. Kink RD
3. Fuzzy RD: Change in treatment probability less than 1
4. Fuzzy Kink RD
5. RDiT: running variable is time.

Others:

- Multiple cutoff
- Multiple Scores
- Geographic RD
- Dynamic Treatments
- Continuous Treatments

Consider

$$D_i = 1_{X_i > c}$$

$$D_i = \begin{cases} D_i = 1 \text{ if } X_i > C \\ D_i = 0 \text{ if } X_i < C \end{cases}$$

where

- D_i = treatment effect
- X_i = score variable (continuous)
- c = cutoff point

Identification (Identifying assumptions) of RD:

Average Treatment Effect at the cutoff (Continuity-based)

$$\begin{aligned} \alpha_{SRDD} &= E[Y_{1i} - Y_{0i} | X_i = c] \\ &= E[Y_{1i} | X_i = c] - E[Y_{0i} | X_i = c] \\ &= \lim_{x \to c^+} E[Y_{1i} | X_i = c] - \lim_{x \to c^=} E[Y_{0i} | X_i = c] \end{aligned}$$

Average Treatment Effect in a neighborhood (Local Randomization-based):

$$\begin{aligned}\alpha_{LR} &= E[Y_{1i} - Y_{0i}|X_i \in W] \\ &= \frac{1}{N_1}\sum_{X_i \in W, T_i=1} Y_i - \frac{1}{N_0}\sum_{X_i \in W, T_i=0} Y_i\end{aligned}$$

RDD estimates the local average treatment effect (LATE), at the cutoff point which is not at the individual or population levels.

Since researchers typically care more about the internal validity, than external validity, localness affects only external validity.

Assumptions:

- Independent assignment
- Continuity of conditional regression functions
  - $E[Y(0)|X = x]$ and $E[Y(1)|X = x]$ are continuous in x.
- RD is valid if cutpoint is **exogenous (i.e., no endogenous selection)** and running variable is **not manipulable**
- Only treatment(s) (e.g., could be joint distribution of multiple treatments) cause discontinuity or jump in the outcome variable
- All other factors are **smooth** through the cutoff (i.e., threshold) value. (we can also test this assumption by seeing no discontinuity in other factors). If they "jump", they will bias your causal estimate

Threats to RD

- Variables (other than treatment) change discontinuously at the cutoff
  - We can test for jumps in these variables (including pre-treatment outcome)
- Multiple discontinuities for the assignment variable
- Manipulation of the assignment variable
  - At the cutoff point, check for continuity in the density of the assignment variable.

27.1 Estimation and Inference

27.1.1 Local Randomization-based

Additional Assumption: Local Randomization approach assumes that inside the chosen window $W = [c - w, c + w]$ are assigned to treatment as good as random:

1. Joint probability distribution of scores for units inside the chosen window W is known

2. Potential outcomes are not affected by value of the score

This approach is stronger than the Continuity-based because we assume the regressions are continuously at c and unaffected by the running variable within window W

Because we can choose the window W (within which random assignment is plausible), the sample size can typically be small.

To choose the window W, we can base on either

1. where the pre-treatment covariate-balance is observed
2. independent tests between outcome and score
3. domain knowledge

To make inference, we can either use

- (Fisher) randomization inference
- (Neyman) design-based

27.1.2 Continuity-based

- also known as the local polynomial method
  - as the name suggests, global polynomial regression is not recommended (because of lack of robustness, and over-fitting and Runge's phenomenon)

Step to estimate local polynomial regression

1. Choose polynomial order and weighting scheme
2. Choose bandwidth that has optimal MSE or coverage error
3. Estimate the parameter of interest
4. Examine robust bias-correct inference

27.2 Specification Checks

1. Balance Checks
2. Sorting/Bunching/Manipulation
3. Placebo Tests
4. Sensitivity to Bandwidth Choice

27.2.1 Balance Checks

- Also known as checking for Discontinuities in Average Covariates
- Null Hypothesis: The average effect of covariates on pseudo outcomes (i.e., those qualitatively cannot be affected by the treatment) is 0.

- If this hypothesis is rejected, you better have a good reason to why because it can cast serious doubt on your RD design.

27.2.2 Sorting/Bunching/Manipulation

- Also known as checking for A Discontinuity in the Distribution of the Forcing Variable
- Also known as clustering or density test
- Formal test is McCrary sorting test (McCrary, 2008) or (Cattaneo et al., 2019)
- Since human subjects can manipulate the running variable to be just above or below the cutoff (assuming that the running variable is manipulable), especially when the cutoff point is known in advance for all subjects, this can result in a discontinuity in the distribution of the running variable at the cutoff (i.e., we will see "bunching" behavior right before or after the cutoff)>
  - People would like to sort into treatment if it's desirable. The density of the running variable would be 0 just below the threshold
  - People would like to be out of treatment if it's undesirable
- (McCrary, 2008) proposes a density test (i.e., a formal test for manipulation of the assignment variable).
  - H_0: The continuity of the density of the running variable (i.e., the covariate that underlies the assignment at the discontinuity point)
  - H_a: A jump in the density function at that point
  - Even though it's not a requirement that the density of the running must be continuous at the cutoff, but a discontinuity can suggest manipulations.
- (Zhang and Rubin, 2003; Lee, 2009; Aronow et al., 2019) offers a guide to know when you should warrant the manipulation
- Usually it's better to know your research design inside out so that you can suspect any manipulation attempts.
  - We would suspect the direction of the manipulation. And typically, it's one-way manipulation. In cases where we might have both ways, theoretically they would cancel each other out.
- We could also observe partial manipulation in reality (e.g., when subjects can only imperfectly manipulate). But typically, as we treat it like fuzzy RD, we would not have identification problems. But complete manipulation would lead to serious identification issues.

- Remember: even in cases where we fail to reject the null hypothesis for the density test, we could not rule out completely that identification problem exists (just like any other hypotheses)
- Bunching happens when people self-select to a specific value in the range of a variable (e.g., key policy thresholds).
- Review paper (Kleven, 2016)
- **This test can only detect manipulation that changes the distribution of the running variable**. If you can choose the cutoff point or you have 2-sided manipulation, this test will fail to detect it.
- Histogram in bunching is similar to a density curve (we want narrower bins, wider bins bias elasticity estimates)
- We can also use bunching method to study individuals' or firm's responsiveness to changes in policy.
- Under RD, we assume that we don't have any manipulation in the running variable. However, bunching behavior is a manipulation by firms or individuals. Thus, violating this assumption.
  - Bunching can fix this problem by estimating what densities of individuals would have been without manipulation (i.e., manipulation-free counterfactual).
  - **The fraction of persons who manipulated** is then calculated by comparing the observed distribution to manipulation-free counterfactual distributions.
  - Under RD, we do not need this step because the observed and manipulation-free counterfactual distributions are assumed to be the same. RD assume there is no manipulation (i.e., assume the manipulation-free counterfactual distribution)

When running variable and outcome variable are simultaneously determined, we can use a modified RDD estimator to have consistent estimate. (Bajari et al., 2011)

- **Assumptions**:
  - Manipulation is **one-sided**: People move one way (i.e., either below the threshold to above the threshold or vice versa, but not to or away the threshold), which is similar to the monotonicity assumption under instrumental variable 36.1.3.1
  - Manipulation is **bounded** (also known as regularity assumption): so that we can use people far away from this threshold to derive at our counterfactual distribution [Blomquist et al. (2021)](Bertanha et al., 2021)

Steps:

1. Identify the window in which the running variable contains bunching behavior. We can do this step empirically based on Bosch et al. (2020). Additionally robustness test is needed (i.e., varying the manipulation window).
2. Estimate the manipulation-free counterfactual
3. Calculating the standard errors for inference can follow (Chetty et al., 2016) where we bootstrap re-sampling residuals in the estimation of the counts of individuals within bins (large data can render this step unnecessary).

If we pass the bunching test, we can move on to the Placebo Test

McCrary (2008) test

A jump in the density at the threshold (i.e., discontinuity) hold can serve as evidence for sorting around the cutoff point

```
library(rdd)

# you only need the runing variable and the cutoff point

# Example by the package's authors
#No discontinuity
x<-runif(1000,-1,1)
DCdensity(x,0)
```

```
#> [1] 0.2428087

#Discontinuity
```

```
x<-runif(1000,-1,1)
x<-x+2*(runif(1000,-1,1)>0&x<0)
DCdensity(x,0)
```

```
#> [1] 0.0465357
```

Cattaneo et al. (2019) test

```r
library(rddensity)

# Example by the package's authors
# Continuous Density
set.seed(1)
x <- rnorm(2000, mean = -0.5)
rdd <- rddensity(X = x, vce = "jackknife")
summary(rdd)
#>
#> Manipulation testing using local polynomial density estimation.
#>
#> Number of obs =       2000
#> Model =               unrestricted
#> Kernel =              triangular
#> BW method =           estimated
#> VCE method =          jackknife
#>
#> c = 0                 Left of c           Right of c
#> Number of obs         1376                624
#> Eff. Number of obs    354                 345
```

```
#> Order est. (p)        2                   2
#> Order bias (q)        3                   3
#> BW est. (h)           0.514               0.609
#>
#> Method                T                   P > |T|
#> Robust                -0.6798             0.4966
#>
#>
#> P-values of binomial tests (H0: p=0.5).
#>
#> Window Length / 2          <c     >=c    P>|T|
#> 0.036                      28      20    0.3123
#> 0.072                      46      39    0.5154
#> 0.107                      68      59    0.4779
#> 0.143                      94      79    0.2871
#> 0.179                     122     103    0.2301
#> 0.215                     145     130    0.3986
#> 0.250                     163     156    0.7370
#> 0.286                     190     176    0.4969
#> 0.322                     214     200    0.5229
#> 0.358                     249     218    0.1650

# you have to specify your own plot (read package manual)
```

27.2.3 Placebo Tests

- Also known as Discontinuities in Average Outcomes at Other Values
- We should not see any jumps at other values (either $X_i < c$ or $X_i \geq c$)
  - Use the same bandwidth you use for the cutoff, and move it along the running variable: testing for a jump in the conditional mean of the outcome at the median of the running variable.
- Also known as falsification checks
- Before and after the cutoff point, we can run the placebo test to see whether X's are different).
- The placebo test is where you expect your coefficients to be not different from 0.
- This test can be used for
  - Testing no discontinuity in predetermined variables:
  - Testing other discontinuities

  - Placebo outcomes: we should see any changes in other outcomes that shouldn't have changed.
  - Inclusion and exclusion of covariates: RDD parameter estimates should not be sensitive to the inclusion or exclusion of other covariates.
- This is analogous to Experimental Design where we cannot only test whether the observables are similar in both treatment and control groups (if we reject this, then we don't have random assignment), but we cannot test unobservables.

Balance on observable characteristics on both sides

$$Z_i = \alpha_0 + \alpha_1 f(x_i) + [I(x_i \geq c)]\alpha_2 + [f(x_i) \times I(x_i \geq c)]\alpha_3 + u_i$$

where

- x_i is the running variable
- Z_i is other characteristics of people (e.g., age, etc)

Theoretically, Z_i should no be affected by treatment. Hence, $E(\alpha_2) = 0$

Moreover, when you have multiple Z_i, you typically have to simulate joint distribution (to avoid having significant coefficient based on chance).

The only way that you don't need to generate joint distribution is when all Z_i's are independent (unlikely in reality).

Under RD, you shouldn't have to do any Matching Methods. Because just like when you have random assignment, there is no need to make balanced dataset before and after the cutoff. If you have to do balancing, then your RD assumptions are probably wrong in the first place.

27.2.4 Sensitivity to Bandwidth Choice

- Methods for bandwidth selection
  - Ad-hoc or substantively driven
  - Data driven: cross validation
  - Conservative approach: (Calonico et al., 2020)
- The objective is to minimize the mean squared error between the estimated and actual treatment effects.
- Then, we need to see how sensitive our results will be dependent on the choice of bandwidth.
- In some cases, the best bandwidth for testing covariates may not be the best bandwidth for treating them, but it may be close.

```r
# find optimal bandwidth by Imbens-Kalyanaraman
rdd::IKbandwidth(running_var,
                 outcome_var,
                 cutpoint = "",
                 kernel = "triangular") # can also pick other kernels
```

27.2.5 Manipulation Robust Regression Discontinuity Bounds

- McCrary (2008) linked density jumps at cutoffs in RD studies to potential manipulation.
  - If no jump is detected, researchers proceed with RD analysis; if detected, they halt using the cutoff for inference.
  - Some studies use the "doughnut-hole" method, excluding near-cutoff observations and extrapolating, which contradicts RD principles.
    - False negative could be due to a small sample size and can lead to biased estimates, as units near the cutoff may still differ in unobserved ways.
    - Even correct rejections of no manipulation may overlook that the data can still be informative despite modest manipulation.
    - Gerard et al. (2020) introduces a systematic approach to handle potentially manipulated variables in RD designs, addressing both concerns.
- The model introduces two types of unobservable units in RD designs:
  - **always-assigned** units, which are always on one side of the cutoff,
  - **potentially-assigned** units, which fit traditional RD assumptions.
    - The standard RD model is a subset of this broader model, which assumes no always-assigned units.
- Identifying assumption: manipulation occurs through one-sided selection.
- The approach does not make a binary decision on manipulation in RD designs but assesses its extent and worst-case impact.

Two steps are used:

1. Determining the proportion of always-assigned units using the discontinuity at the cutoff
2. Bounding treatment effects based on the most extreme feasible outcomes for these units.

- For sharp RD designs, bounds are established by trimming extreme outcomes near the cutoff; for fuzzy designs, the process involves more complex adjustments due to additional model constraints.
- Extensions of the study use covariate information and economic behavior assumptions to refine these bounds and identify covariate distributions among unit types at the cutoff.

Setup

Independent data points (X_i, Y_i, D_i), where X_i is the running variable, Y_i is the outcome, and D_i indicates treatment status (1 if treated, 0 otherwise). Treatment is assigned based on $X_i \geq c$.

The design is *sharp* if $D_i = I(X_i \geq c)$ and *fuzzy* otherwise.

The population is divided into:

- **Potentially-assigned units** ($M_i = 0$): Follow the standard RD framework, with potential outcomes $Y_i(d)$ and potential treatment states $D_i(x)$.
- **Always-assigned units** ($M_i = 1$): These units do not require potential outcomes or states, and always have X_i values beyond the cutoff.

Assumptions

1. **Local Independence and Continuity**:
   - $P(D = 1|X = c^+, M = 0) > P(D = 1|X = c^-, M = 0)$
   - No defiers: $P(D^+ \geq D^-|X = c, M = 0) = 1$
   - Continuity in potential outcomes and states at c.
   - $F_{X|M=0}(x)$ is differentiable at c, with a positive derivative.
2. **Smoothness of the Running Variable among Potentially-Assigned Units**:
   - The derivative of $F_{X|M=0}(x)$ is continuous at c.
3. **Restrictions on Always-Assigned Units**:
   - $P(X \geq c|M = 1) = 1$ and $F_{X|M=1}(x)$ is right-differentiable (or left-differentiable) at c.
   - This (local) one-sided manipulation assumption allows identification of the proportion of always-assigned units among all units close to the cutoff.

When always-assigned unit exist, the RD design is fuzzy because we have

1. Treated and untreated units among the potentially-assigned (below and above the cutoff)
2. Always-assigned units (above the cutoff).

Causal Effects of Interest

causal effects among potentially-assigned units:

$$\Gamma = E[Y(1) - Y(0)|X = c, D^+ > D^-, M = 0]$$

This parameter represents the local average treatment effect (LATE) for the subgroup of "compliers"—units that receive treatment if and only if their running variable X_i exceeds a certain cutoff.

The parameter Γ captures the causal effect of changes in the cutoff level on treatment status among potentially-assigned compliers.

RD designs with a manipulated running variable	**"Doughnut-Hole" RD Designs**:
• Focuses on actual observations at the cutoff, not hypothetical true values. • Provides a direct and observable estimate of causal effects, without reliance on hypothetical constructs.	• Exclude observations around the cutoff and use extrapolation from the trends outside this excluded range to infer causal effects at the cutoff • Assumes a hypothetical population existing in a counterfactual scenario without manipulation. • Requires strong assumptions about the nature of manipulation and the minimal impact of extrapolation biases.

Identification of τ in RD Designs

- Identification challenges arise due to the inability to distinguish always-assigned from potentially-assigned units, thus Γ is not point identified. We establish sharp bounds on Γ
- These bounds are supported by the stochastic dominance of the potential outcome CDFs over observed distributions.

Unit Types and Notation:

- C_0: Potentially-assigned compliers.
- A_0: Potentially-assigned always-takers.
- N_0: Potentially-assigned never-takers.
- T_1: Always-assigned treated units.
- U_1: Always-assigned untreated units.

The measure τ , representing the proportion of always-assigned units near the cutoff, is point identified by the discontinuity in the observed running variable density f_X at the cutoff

Sharp RD:

- Units to the left of the cutoff are potentially assigned units. The distribution of their observed outcomes (Y) are the outcomes $Y(0)$ of potentially-assigned compliers (C_0) at the cutoff.

- To determine the bounds on the treatment effect (Γ), we need to assess the distribution of treated outcomes ($Y(1)$) for the same potentially-assigned compliers at the cutoff.
- Information regarding the treated outcomes ($Y(1)$) comes exclusively from the subpopulation of treated units, which includes both potentially-assigned compliers (C_0) and those always assigned units (T_1).
- With τ point identified, we can estimate sharp bounds on Γ.

Fuzzy RD:

Table 27.2: Note: Table on page 848 (Gerard et al., 2020)

Subpopulation	Types of units
$X = c^+, D = 1$	C_0, A_0, T_1
$X = c^-, D = 1$	A_0
$X = c^+, D = 0$	N_0, U_1
$X = c^-, D = 0$	C_0, N_0

- **Unit Types and Combinations**: There are five distinct unit types and four combinations of treatment assignments and decisions relevant to the analysis. These distinctions are important because they affect how potential outcomes are analyzed and bounded.
- **Outcome Distributions**: The analysis involves estimating the distribution of potential outcomes (both treated and untreated) among potentially-assigned compliers at the cutoff.
- **Three-Step Process**:
  1. **Potential Outcomes Under Treatment**: Bounds on the distribution of treated outcomes are determined using data from treated units.
  2. **Potential Outcomes Under Non-Treatment**: Bounds on the distribution of untreated outcomes are derived using data from untreated units.
  3. **Bounds on Parameters of Interest**: Using the bounds from the first two steps, sharp upper and lower bounds on the local average treatment effect are derived.
- **Extreme Value Consideration**: The bounds for treatment effects are based on "extreme" scenarios under worst-case assumptions about the distribution of potential outcomes, making them sharp but empirically relevant within the data constraints.

Extensions:

- **Quantile Treatment Effects**: alternative to average effects by focusing on different quantiles of the outcome distribution, which are less affected by extreme values.
- **Applicability to Discrete Outcomes**
- **Behavioral Assumptions Impact**: Assuming a high likelihood of treatment among always-assigned units can narrow the bounds of treatment effects by refining the analysis of potential outcomes.
- **Utilization of Covariates**: Incorporating covariates measured prior to treatment can refine the bounds on treatment effects and help target policies by identifying covariate distributions among different unit types.

Notes:

- Quantile Treatment Effects (QTEs): QTE bounds are less sensitive to the tails of the outcome distribution, making them tighter than ATE bounds.
  - Inference on ATEs is sensitive to the extent of manipulation, with confidence intervals widening significantly with small degrees of assumed manipulation.
  - Inference on QTEs is less affected by manipulation, remaining meaningful even with larger degrees of manipulation.
- Alternative Inference Strategy when manipulation is believed to be unlikely. Try different hypothetical values of τ

```
devtools::install_github("francoisgerard/rdbounds/R")
```

```
library(formattable)
library(data.table)
library(rdbounds)
set.seed(123)
df <- rdbounds_sampledata(1000, covs = FALSE)
#> [1] "True tau: 0.117999815082062"
#> [1] "True treatment effect on potentially-assigned: 2"
#> [1] "True treatment effect on right side of cutoff: 2.35399944524618"
head(df)
#>            x         y treatment
#> 1 -1.2532616  2.684827         0
#> 2 -0.5146925  5.845219         0
#> 3  3.4853777  6.166070         0
#> 4  0.1576616  3.227139         0
#> 5  0.2890962  7.031685         1
#> 6  3.8350019 10.238570         1

rdbounds_est <-
    rdbounds(
```

```
        y = df$y,
        x = df$x,
        # covs = as.factor(df$cov),
        treatment = df$treatment,
        c = 0,
        discrete_x = FALSE,
        discrete_y = FALSE,
        bwsx = c(.2, .5),
        bwy = 1,

        # for median effect use
        # type = "qte",
        # percentiles = .5,

        kernel = "epanechnikov",
        orders = 1,
        evaluation_ys = seq(from = 0, to = 15, by = 1),
        refinement_A = TRUE,
        refinement_B = TRUE,
        right_effects = TRUE,
        yextremes = c(0, 15),
        num_bootstraps = 5
    )
#> [1] "The proportion of always-assigned units just to the right of the cutoff is estimated to b
#> [1] "2025-02-08 18:44:40.297241 Estimating CDFs for point estimates"
#> [1] "2025-02-08 18:44:40.51386 .....Estimating CDFs for units just to the right of the cutoff"
#> [1] "2025-02-08 18:44:42.550442 Estimating CDFs with nudged tau (tau_star)"
#> [1] "2025-02-08 18:44:42.587801 .....Estimating CDFs for units just to the right of the cutoff
#> [1] "2025-02-08 18:44:45.485849 Beginning parallelized output by bootstrap.."
#> [1] "2025-02-08 18:44:49.357832 Computing Confidence Intervals"
#> [1] "2025-02-08 18:45:02.269842 Time taken:0.37 minutes"
```

```
rdbounds_summary(rdbounds_est, title_prefix = "Sample Data Results")
#> [1] "Time taken: 0.37 minutes"
#> [1] "Sample size: 1000"
#> [1] "Local Average Treatment Effect:"
#> $tau_hat
#> [1] 0.3804665
#>
#> $tau_hat_CI
#> [1] 0.4180576 1.4333618
#>
#> $takeup_increase
#> [1] 0.6106973
#>
```

```
#> $takeup_increase_CI
#> [1] 0.4332132 0.7881814
#>
#> $TE_SRD_naive
#> [1] 1.589962
#>
#> $TE_SRD_naive_CI
#> [1] 1.083977 2.095948
#>
#> $TE_SRD_bounds
#> [1] 0.7165801 2.3454868
#>
#> $TE_SRD_CI
#> [1] -1.200992  7.733214
#>
#> $TE_SRD_covs_bounds
#> [1] NA NA
#>
#> $TE_SRD_covs_CI
#> [1] NA NA
#>
#> $TE_FRD_naive
#> [1] 2.583271
#>
#> $TE_FRD_naive_CI
#> [1] 1.506573 3.659969
#>
#> $TE_FRD_bounds
#> [1] 1.273896 3.684860
#>
#> $TE_FRD_CI
#> [1] -1.266341 10.306116
#>
#> $TE_FRD_bounds_refinementA
#> [1] 1.273896 3.684860
#>
#> $TE_FRD_refinementA_CI
#> [1] -1.266341 10.306116
#>
#> $TE_FRD_bounds_refinementB
#> [1] 1.420594 3.677963
#>
#> $TE_FRD_refinementB_CI
#> [1] NA NA
#>
```

```
#> $TE_FRD_covs_bounds
#> [1] NA NA
#>
#> $TE_FRD_covs_CI
#> [1] NA NA
#>
#> $TE_SRD_CIs_manipulation
#> [1] NA NA
#>
#> $TE_FRD_CIs_manipulation
#> [1] NA NA
#>
#> $TE_SRD_right_bounds
#> [1] -2.056178  3.650820
#>
#> $TE_SRD_right_CI
#> [1] -11.044937   9.282217
#>
#> $TE_FRD_right_bounds
#> [1] -2.900703  5.272370
#>
#> $TE_FRD_right_CI
#> [1] -14.75421  15.58640
```

```
rdbounds_est_tau <-
    rdbounds(
        y = df$y,
        x = df$x,
        # covs = as.factor(df$cov),
        treatment = df$treatment,
        c = 0,
        discrete_x = FALSE,
        discrete_y = FALSE,
        bwsx = c(.2, .5),
        bwy = 1,
        kernel = "epanechnikov",
        orders = 1,
        evaluation_ys = seq(from = 0, to = 15, by = 1),
        refinement_A = TRUE,
        refinement_B = TRUE,
        right_effects = TRUE,
        potential_taus = c(.025, .05, .1, .2),
        yextremes = c(0, 15),
        num_bootstraps = 5
    )
```

```
#> [1] "The proportion of always-assigned units just to the right of the cutoff is est
#> [1] "2025-02-08 18:45:02.886817 Estimating CDFs for point estimates"
#> [1] "2025-02-08 18:45:03.07515 .....Estimating CDFs for units just to the right of
#> [1] "2025-02-08 18:45:04.85827 Estimating CDFs with nudged tau (tau_star)"
#> [1] "2025-02-08 18:45:04.900485 .....Estimating CDFs for units just to the right of
#> [1] "2025-02-08 18:45:07.925693 Beginning parallelized output by bootstrap.."
#> [1] "2025-02-08 18:45:11.854381 Estimating CDFs with fixed tau value of: 0.025"
#> [1] "2025-02-08 18:45:11.920705 Estimating CDFs with fixed tau value of: 0.05"
#> [1] "2025-02-08 18:45:11.960668 Estimating CDFs with fixed tau value of: 0.1"
#> [1] "2025-02-08 18:45:12.001689 Estimating CDFs with fixed tau value of: 0.2"
#> [1] "2025-02-08 18:45:13.004577 Beginning parallelized output by bootstrap x fixed
#> [1] "2025-02-08 18:45:16.838179 Computing Confidence Intervals"
#> [1] "2025-02-08 18:45:29.845546 Time taken:0.45 minutes"
```

```
causalverse::plot_rd_aa_share(rdbounds_est_tau) # For SRD (default)
```

```
# causalverse::plot_rd_aa_share(rdbounds_est_tau, rd_type = "FRD")  # For FRD
```

27.3 Fuzzy RD Design

When you have cutoff that does not perfectly determine treatment, but creates a discontinuity in the likelihood of receiving the treatment, you need another instrument

For those that are close to the cutoff, we create an instrument for D_i

$$Z_i = \begin{cases} 1 & \text{if } X_i \geq c \\ 0 & \text{if } X_c < c \end{cases}$$

Then, we can estimate the effect of the treatment for compliers only (i.e., those treatment D_i depends on Z_i)

The LATE parameter

$$\lim_{c-\epsilon \leq X \leq c+\epsilon, \epsilon \to 0} \left(\frac{E(Y|Z=1) - E(Y|Z=0)}{E(D|Z=1) - E(D|Z=0)} \right)$$

equivalently, the canonical parameter:

$$\frac{lim_{x \downarrow c} E(Y|X=x) - \lim_{x \uparrow c} E(Y|X=x)}{\lim_{x \downarrow c} E(D|X=x) - \lim_{x \uparrow c} E(D|X=x)}$$

Two equivalent ways to estimate

1. First
    1. Sharp RDD for Y
    2. Sharp RDD for D
    3. Take the estimate from step 1 divide by that of step 2
2. Second: Subset those observations that are close to c and run instrumental variable Z

27.4 Regression Kink Design

- If the slope of the treatment intensity changes at the cutoff (instead of the level of treatment assignment), we can have regression kink design
- Example: unemployment benefits

Sharp Kink RD parameter

$$\alpha_{KRD} = \frac{\lim_{x \downarrow c} \frac{d}{dx} E[Y_i|X_i = x] - \lim_{x \uparrow c} \frac{d}{dx} E[Y_i|X_i = x]}{\lim_{x \downarrow c} \frac{d}{dx} b(x) - \lim_{x \uparrow c} \frac{d}{dx} b(x)}$$

where $b(x)$ is a known function inducing "kink"

Fuzzy Kink RD parameter

$$\alpha_{KRD} = \frac{\lim_{x \downarrow c} \frac{d}{dx} E[Y_i|X_i = x] - \lim_{x \uparrow c} \frac{d}{dx} E[Y_i|X_i = x]}{\lim_{x \downarrow c} \frac{d}{dx} E[D_i|X_i = x] - \lim_{x \uparrow c} \frac{d}{dx} E[D_i|X_i = x]}$$

27.5 Multi-cutoff

$$\tau(x, c) = E[Y_{1i} - Y_{0i}|X_i = x, C_i = c]$$

27.6 Multi-score

Multi-score (in multiple dimensions) (e.g., math and English cutoff for certain honor class):

$$\tau(x_1, x_2) = E[Y_{1i} - Y_{0i}|X_{1i} = x_1, X_{2i} = x]$$

27.7 Steps for Sharp RD

1. Graph the data by computing the average value of the outcome variable over a set of bins (large enough to see a smooth graph, and small enough to make the jump around the cutoff clear).
2. Run regression on both sides of the cutoff to get the treatment effect
3. Robustness checks:
    1. Assess possible jumps in other variables around the cutoff
    2. Hypothesis testing for bunching
    3. Placebo tests
    4. Varying bandwidth

27.8 Steps for Fuzzy RD

1. Graph the data by computing the average value of the outcome variable over a set of bins (large enough to see a smooth graph, and small enough to make the jump around the cutoff clear).
2. Graph the probability of treatment
3. Estimate the treatment effect using 2SLS
4. Robustness checks:
    1. Assess possible jumps in other variables around the cutoff
    2. Hypothesis testing for bunching
    3. Placebo tests
    4. Varying bandwidth

27.9 Steps for RDiT (Regression Discontinuity in Time)

Notes:

- Additional assumption: Time-varying confounders change smoothly across the cutoff date
- Typically used in policy implementation in the same date for all subjects, but can also be used for cases where implementation dates are different between subjects. In the second case, researchers typically use different RDiT specification for each time series.
- Sometimes the date of implementation is not randomly assigned by chosen strategically. Hence, RDiT should be thought of as the "discontinuity at a threshold" interpretation of RD (not as "local randomization"). (Hausman and Rapson, 2018, p. 8)
- Normal RD uses variation in the N dimension, while RDiT uses variation in the T dimension
- Choose polynomials based on BIC typically. And can have either global polynomial or pre-period and post-period polynomial for each time series (but usually the global one will perform better)
- Could use **augmented local linear** outlined by (Hausman and Rapson, 2018, p. 12), where estimate the model with all the control first then take the residuals to include in the model with the RDiT treatment (remember to use bootstrapping method to account for the first-stage variance in the second stage).

Pros:

- can overcome cases where there is no cross-sectional variation in treatment implementation (DID is not feasible)
  - There are papers that use both RDiT and DID to (1) see the differential treatment effects across individuals/ space (Auffhammer and Kellogg, 2011) or (2) compare the 2 estimates where the control group's validity is questionable (Gallego et al., 2013).
- Better than pre/post comparison because it can include flexible controls
- Better than event studies because it can use long-time horizons (may not be too relevant now since the development long-time horizon event studies), and it can use higher-order polynomials time control variables.

Cons:

- Taking observation for from the threshold (in time) can bias your estimates because of unobservables and time-series properties of the data generating process.
- (McCrary, 2008) test is not possible (see Sorting/Bunching/Manipulation) because when the density of the running (time) is uniform, you can't use

the test.

- Time-varying unobservables may impact the dependent variable discontinuously
- Error terms are likely to include persistence (serially correlated errors)
- Researchers cannot model time-varying treatment under RDiT
  - In a small enough window, the local linear specification is fine, but the global polynomials can either be too big or too small (Hausman and Rapson, 2018)

Biases

- Time-Varying treatment Effects
  - increase sample size either by
    * more granular data (greater frequency): will not increase power because of the problem of serial correlation
    * increasing time window: increases bias from other confounders
  - 2 additional assumption:
    * Model is correctly specified (with all confoudners or global polynomial approximation)
    * Treatment effect is correctly specified (whether it's smooth and constant, or varies)
    * These 2 assumptions do not interact (we don't want them to interact - i.e., we don't want the polynomial correlated with the unobserved variation in the treatment effect)
  - There usually a difference between short-run and long-run treatment effects, but it's also possibly that the bias can stem from the over-fitting problem of the polynomial specification. (Hausman and Rapson, 2018, p. 544)
- Autoregression (serial dependence)
  - Need to use **clustered standard errors** to account for serial dependence in the residuals
  - In the case of serial dependence in ϵ_{it}, we don't have a solution, including a lagged dependent variable would misspecify the model (probably find another research project)
  - In the case of serial dependence in y_{it}, with long window, it becomes fuzzy to what you try to recover. You can include the **lagged dependent variable** (bias can still come from the time-varying treatment or over-fitting of the global polynomial)

- Sorting and Anticipation Effects
  - Cannot run the (McCrary, 2008) because the density of the time running variable is uniform
  - Can still run tests to check discontinuities in other covariates (you want no discontinuities) and discontinuities in the outcome variable at other placebo thresholds (you don't want discontinuities)
  - Hence, it's hard to argue for the causal effect here because it could be the total effect of the causal treatment and the unobserved sorting/anticipation/adaptation/avoidance effects. You can only argue that there is no such behavior

Recommendations for robustness check following (Hausman and Rapson, 2018, p. 549)

1. Plot the raw data and residuals (after removing confounders or trend). With varying polynomial and local linear controls, inconsistent results can be a sign of time-varying treatment effects.
2. Using global polynomial, you could overfit, then show polynomial with different order and alternative local linear bandwidths. If the results are consistent, you're okay
3. Placebo Tests: estimate another RD (1) on another location or subject (that did not receive the treatment) or (2) use another date.
4. Plot RD discontinuity on continuous controls
5. Donut RD to see if avoiding the selection close to the cutoff would yield better results (Barreca et al., 2011)
6. Test for auto-regression (using only pre-treatment data). If there is evidence for autoregression, include the lagged dependent variable
7. Augmented local linear (no need to use global polynomial and avoid overfitting)
   1. Use full sample to exclude the effect of important predictors
   2. Estimate the conditioned second stage on a smaller sample bandwidth

Examples from (Hausman and Rapson, 2018, p. 534) in

econ

- (Davis, 2008): Air quality
- (Auffhammer and Kellogg, 2011): Air quality
- (Chen et al., 2018): Air quality
- (De Paola et al., 2013): car accidents
- (Gallego et al., 2013): air quality
- (Bento et al., 2014): Traffic
- (Anderson, 2014): Traffic

- (Burger et al., 2014): Car accidents
- (Brodeur et al., 2021): Covid19 lock-downs on well-being

marketing

- (Busse et al., 2006, Busse et al. (2013)): Vehicle prices
- (Chen et al., 2009a): Customer Satisfaction
- (Busse et al., 2010): Vehicle prices
- (Davis and Kahn, 2010): vehicle prices

27.10 Evaluation of an RD

- Evidence for (either formal tests or graphs)
  - Treatment and outcomes change discontinuously at the cutoff, while other variables and pre-treatment outcomes do not.
  - No manipulation of the assignment variable.
- Results are robust to various functional forms of the forcing variable
- Is there any other (unobserved) confound that could cause the discontinuous change at the cutoff (i.e., multiple forcing variables / bundling of institutions)?
- External Validity: How likely the result at the cutoff will generalize?

General Model

$$Y_i = \beta_0 + f(x_i)\beta_1 + [I(x_i \geq c)]\beta_2 + \epsilon_i$$

where $f(x_i)$ is any functional form of x_i

Simple case

When $f(x_i) = x_i$ (linear function)

$$Y_i = \beta_0 + x_i\beta_1 + [I(x_i \geq c)]\beta_2 + \epsilon_i$$

RD gives you β_2 (causal effect) of X on Y at the cutoff point

In practice, everyone does

$$Y_i = \alpha_0 + f(x)\alpha_1 + [I(x_i \geq c)]\alpha_2 + [f(x_i) \times [I(x_i \geq c)]\alpha_3 + u_i$$

where we estimate different slope on different sides of the line

and if you estimate α_3 to be no different from 0 then we return to the simple case

Notes:

- Sparse data can make α_3 large differential effect
- People are very skeptical when you have complex $f(x_i)$, usual simple function forms (e.g., linear, squared term, etc.) should be good. However, if you still insist, then **non-parametric estimation** can be your best bet.

Bandwidth of c (window)

- Closer to c can give you lower bias, but also efficiency
- Wider c can increase bias, but higher efficiency.
- Optimal bandwidth is very controversial, but usually we have to do it in the appendix for research article anyway.
- We can either
  - drop observations outside of bandwidth or
  - weight depends on how far and close to c

27.11 Applications

Examples in marketing:

- (Narayanan and Kalyanam, 2015)
- (Hartmann et al., 2011): nonparametric estimation and guide to identifying causal marketing mix effects

Packages in R (see (Thoemmes et al., 2017) for detailed comparisons): all can handle both sharp and fuzzy RD

- `rdd`
- `rdrobust` estimation, inference and plot
- `rddensity` discontinuity in density tests (Sorting/Bunching/Manipulation) using local polynomials and binomial test
- `rdlocrand` covariate balance, binomial tests, window selection
- `rdmulti` multiple cutoffs and multiple scores
- `rdpower` power, sample selection
- `rddtools`

Package	rdd	rdrobust	rddtools
Coefficient estimator bandwidth selectors	Local linear regression (Imbens and Kalyanaraman, 2012)	local polynomial regression (Calonico et al., 2020) (Imbens and Kalyanaraman, 2012) (Calonico et al., 2014)	local polynomial regression (Imbens and Kalyanaraman, 2012)
Kernel functions	Epanechnikov Gaussian	Epanechnikov	Gaussian
• Triangular			
• Rectangular			
Bias Correction		Local polynomial regression	
Covariate options	Include	Include	Include Residuals
Assumptions testing	McCrary sorting		McCrary sorting Equality of covariates distribution and mean

based on table 1 (Thoemmes et al., 2017) (p. 347)

27.11.1 Example 1

Example by Leihua Ye

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 W_i + u_i$$

$$X_i = \begin{cases} 1, W_i \geq c \\ 0, W_i < c \end{cases}$$

```
#cutoff point = 3.5
GPA <- runif(1000, 0, 4)
future_success <- 10 + 2 * GPA + 10 * (GPA >= 3.5) + rnorm(1000)
#install and load the package 'rddtools'
#install.packages("rddtools")
library(rddtools)
data <- rdd_data(future_success, GPA, cutpoint = 3.5)
```

```r
# plot the dataset
plot(
    data,
    col =  "red",
    cex = 0.1,
    xlab =  "GPA",
    ylab =  "future_success"
)
```

```r
# estimate the sharp RDD model
rdd_mod <- rdd_reg_lm(rdd_object = data, slope =  "same")
summary(rdd_mod)
#>
#> Call:
#> lm(formula = y ~ ., data = dat_step1, weights = weights)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -3.4147 -0.6768 -0.0202  0.6705  3.3654
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 17.09347    0.06770  252.48   <2e-16 ***
#> D            9.87709    0.11937   82.74   <2e-16 ***
#> x            2.03202    0.03309   61.42   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#>
#> Residual standard error: 0.9965 on 997 degrees of freedom
#> Multiple R-squared:  0.9606, Adjusted R-squared:  0.9606
#> F-statistic: 1.217e+04 on 2 and 997 DF,  p-value: < 2.2e-16
```

```
# plot the RDD model along with binned observations
plot(
    rdd_mod,
    cex = 0.1,
    col =  "red",
    xlab =  "GPA",
    ylab =  "future_success"
)
```

27.11.2 Example 2

Bowblis and Smith (2021)

Occupational licensing can either increase or decrease market efficiency:

- More information means more efficiency
- Increased entry barriers (i.e., friction) increase efficiency

Components of RD

- Running variable
- Cutoff: 120 beds or above
- Treatment: you have to have the treatment before the cutoff point.

Under OLS

$$Y_i = \alpha_0 + X_i\alpha_1 + LW_i\alpha_2 + \epsilon_i$$

where

- LW_i Licensed/certified workers (in fraction format for each center).
- Y_i = Quality of service

Bias in α_2

- Mitigation-based: terrible quality can lead to more hiring, which negatively bias α_2
- Preference-based: places that have higher quality staff want to keep high quality staffs.

Under RD

$$\begin{aligned} Y_{ist} = \beta_0 &+ [I(Bed \geq 121)_{ist}]\beta_1 + f(Size_{ist})\beta_2 \\ &+ [f(Size_{ist}) \times I(Bed \geq 121)_{ist}]\beta_3 \\ &+ X_{it}\delta + \gamma_s + \theta_t + \epsilon_{ist} \end{aligned}$$

where

- s = state
- t = year
- i = hospital

This RD is fuzzy

- If right near the threshold (bandwidth), we have states with different sorting (i.e., non-random), then we need the fixed-effect for state s. But then your RD assumption wrong anyway, then you won't do it in the first place
- Technically, we could also run the fixed-effect regression, but because it's lower in the causal inference hierarchy. Hence, we don't do it.
- Moreover, in the RD framework, we don't include t before treatment (but in the FE we have to include before and after)
- If we include π_i for each hospital, then we don't have variation in the causal estimates (because hardly any hospital changes their bed size in the panel)
- When you have β_1 as the intent to treat (because the treatment effect does not coincide with the intent to treat)

- You cannot take those fuzzy cases out, because it will introduce the selection bias.
- Note that we cannot drop cases based on behavioral choice (because we will exclude non-compliers), but we can drop when we have particular behaviors ((e.g., people like round numbers).

Thus, we have to use Instrument variable 36.1.3.1

Stage 1:

$$\begin{aligned} QSW_{ist} = \alpha_0 &+ [I(Bed \geq 121)_{ist}]\alpha_1 + f(Size_{ist})\alpha_2 \\ &+ [f(Size_{ist}) \times I(Bed \geq 121)_{ist}]\alpha_3 \\ &+ X_{it}\delta + \gamma_s + \theta_t + \epsilon_{ist} \end{aligned}$$

(Note: you should have different fixed effects and error term - $\delta, \gamma_s, \theta_t, \epsilon_{ist}$ from the first equation, but I ran out of Greek letters)

Stage 2:

$$\begin{aligned} Y_{ist} = \gamma_0 &+ \gamma_1 Q\hat{W}S_{ist} + f(Size_{ist})\delta_2 \\ &+ [f(Size_{ist}) \times I(Bed \geq 121)]\delta_3 \\ &+ X_{it}\lambda + \eta_s + \tau_t + u_{ist} \end{aligned}$$

- The bigger the jump (discontinuity), the more similar the 2 coefficients ($\gamma_1 \approx \beta_1$) where γ_1 is the average treatment effect (of exposing to the policy)
- β_1 will always be closer to 0 than γ_1
- Figure 1 shows bunching at every 5 units cutoff, but 120 is still out there.
- If we have manipulable bunching, there should be decrease at 130
- Since we have limited number of mass points (at the round numbers), we should clustered standard errors by the mass point

27.11.3 Example 3

Replication of (Carpenter and Dobkin, 2009) by Philipp Leppert, dataset from here

27.11.4 Example 4

For a detailed application, see (Thoemmes et al., 2017) where they use `rdd`, `rdrobust`, `rddtools`

Chapter 28

Synthetic Difference-in-Differences

by (Arkhangelsky et al., 2021)

also known as weighted double-differencing estimators

- Setting: Researchers use panel data to study effects of policy changes.
  - Panel data: repeated observations across time for various units.
  - Some units exposed to policy at different times than others.
- Policy changes often aren't random across units or time.
  - Challenge: Observed covariates might not lead to credible conclusions of no confounding (Imbens and Rubin, 2015)
- To estimate the effects, either
  - Difference-in-differences (DID) method widely used in applied economics.
  - Synthetic Control (SC) methods offer alternative approach for comparative case studies.
- Difference between DID and SC:
  - DID: used with many policy-exposed units; relies on "parallel trends" assumption.
  - SC: used with few policy-exposed units; compensates lack of parallel trends by reweighting units based on pre-exposure trends.
- **New proposition**: Synthetic Difference in Differences (SDID).
  - Combines features of DID and SC.

  - Reweights and matches pre-exposure trends (similar to SC).
  - Invariant to additive unit-level shifts, valid for large-panel inference (like DID).
- Attractive features:
  - SDID provides consistent and asymptotically normal estimates.
  - SDID performs on par with or better than DID in traditional DID settings.
    * where DID can only handle completely random treatment assignment, SDID can handle cases where treatment assignment is correlated with some time or unit latent factors.
  - Similarly, SDID is as good as or better than SC in traditional SC settings.
  - Uniformly random treatment assignment results in unbiased outcomes for all methods, but SDID is more precise.
  - SDID reduces bias effectively for non-uniformly random assignments.
  - SDID's double robustness is akin to the augmented inverse probability weighting estimator (Ben-Michael et al., 2021, Scharfstein et al. (1999)).
  - Very much similar to augmented SC estimator by (Ben-Michael et al., 2021; Arkhangelsky et al., 2021, p. 4112)

Ideal case to use SDID estimator is when

- $N_{ctr} \approx T_{pre}$
- Small T_{post}
- $N_{tr} < \sqrt{N_{ctr}}$

Applications in marketing:

- Lambrecht et al. (2024): TV ads on online browsing and sales.
- Keller et al. (2024): soda tax on marketing effectiveness.

28.1 Understanding

Consider a traditional time-series cross-sectional data

Let Y_{it} denote the outcome for unit i in period t

A balanced panel of N units and T time periods

- $W_{it} \in \{0, 1\}$ is the binary treatment
- N_c never-treated units (control)
- N_t treated units after time T_{pre}

Steps:

1. Find unit weights $\hat{w}^{sdid}$ such that $\sum_{i=1}^{N_c} \hat{w}_i^{sdid} Y_{it} \approx N_t^{-1} \sum_{i=N_c+1}^{N} Y_{it} \forall t = 1, \dots, T_{pre}$ (i.e., pre-treatment trends in outcome of the treated similar to those of control units) (similar to SC).
2. Find time weights $\hat{\lambda}_t$ such that we have a balanced window (i.e., posttreatment outcomes for control units differ consistently from their weighted average pretreatment outcomes).
3. Estimate the average causal effect of treatment

$$(\hat{\tau}^{sdid}, \hat{\mu}, \hat{\alpha}, \hat{\beta}) = \arg \min_{\tau,\mu,\alpha,\beta} \{\sum_{i=1}^{N} \sum_{t=1}^{T} (Y_{it} - \mu - \alpha_i - \beta_t - W_{it}\tau)^2 \hat{w}_i^{sdid} \hat{\lambda}_t^{sdid}\}$$

Better than DiD estimator because τ^{did} does not consider time or unit weights

$$(\hat{\tau}^{did}, \hat{\mu}, \hat{\alpha}, \hat{\beta}) = \arg \min_{\tau,\mu,\alpha,\beta} \{\sum_{i=1}^{N} \sum_{t=1}^{T} (Y_{it} - \mu - \alpha_i - \beta_t - W_{it}\tau)^2\}$$

Better than SC estimator because τ^{sc} lacks unit fixed effete and time weights

$$(\hat{\tau}^{sc}, \hat{\mu}, \hat{\beta}) = \arg \min_{\tau,\mu,\beta} \{\sum_{i=1}^{N} \sum_{t=1}^{T} (Y_{it} - \mu - \beta_t - W_{it}\tau)^2 \hat{w}_i^{sdid}\}$$

	DID	**SC**	**SDID**
Primary Assumption	Absence of intervention leads to parallel evolution across states.	Reweights unexposed states to match pre-intervention outcomes of treated state.	Reweights control units to ensure a parallel time trend with the treated pre-intervention trend.
Reliability Concern	Can be unreliable when pre-intervention trends aren't parallel.	Accounts for non-parallel pre-intervention trends by reweighting.	Uses reweighting to adjust for non-parallel pre-intervention trends.

	DID	SC	SDID
Treatment of Time Periods	All pre-treatment periods are given equal weight.	Doesn't specifically emphasize equal weight for pre-treatment periods.	Focuses only on a subset of pre-intervention time periods, selected based on historical outcomes.
Goal with Reweighting	N/A (doesn't use reweighting).	To match treated state as closely as possible before the intervention.	Make trends of control units parallel (not necessarily identical) to the treated pre-intervention.

Alternatively, think of our parameter of interest as:

$$\hat{\tau} = \hat{\delta}_t - \sum_{i=1}^{N_c} \hat{w}_i \hat{\delta}_i$$

where $\hat{\delta}_t = \frac{1}{N_t} \sum_{i=N_c+1}^{N} \hat{\delta}_i$

Method	Sample Weight	Adjusted outcomes ($\hat{\delta}_i$)	Interpretation
SC	$\hat{w}^{sc} = \min_{w \in R} l_{unit}(w)$	$\frac{1}{T_{post}} \sum_{t=T_{pre}+1}^{T} Y_{it}$	Unweighted treatment period averages
DID	$\hat{w}_i^{did} = N_c^{-1}$	$\frac{1}{T_{post}} \sum_{t=T_{pre}+1}^{T} Y_{it} - \frac{1}{T_{pre}} \sum_{t=1}^{T_{pre}} Y_{it}$	Unweighted differences between average treatment period and pretreatment outcome
SDID	$(\hat{w}_0, \hat{w}^{sdid}) = \min l_{unit}(w_0, w)$	$\frac{1}{T_{post}} \sum_{t=T_{pre}+1}^{T} Y_{it} - \sum_{t=1}^{T_{pre}} \hat{\lambda}_t^{sdid} Y_{it}$	Weighted differences between average treatment period and pretreatment outcome

- The SDID estimator uses weights:
  - Makes two-way fixed effect regression "local."
  - Emphasizes units similar in their past to treated units.
  - Prioritizes periods resembling treated periods.
- Benefits of this localization:
  1. **Robustness**: Using similar units and periods boosts estimator's robustness.

2. **Improved Precision**: Weights can eliminate predictable outcome components.
   - The SEs of SDID are smaller than those of SC and DID
   - Caveat: If there's minor systematic heterogeneity in outcomes, unequal weighting might reduce precision compared to standard DID.

- Weight Design:
  - **Unit Weights**: Makes average outcome for treated units roughly parallel to the weighted average for control units.
  - **Time Weights**: Ensures posttreatment outcomes for control units differ consistently from their weighted average pretreatment outcomes.
- Weights enhance DID's plausibility:
  - Raw data often lacks parallel time trends for treated/control units.
  - Similar techniques (e.g., adjusting for covariates or selecting specific time periods) were used before (Callaway and Sant'Anna, 2021).
  - SDID automates this process, applying a similar logic to weight both units and time periods.
- Time Weights in SDID:
  - Removes bias and boosts precision (i.e., minimizes the influence of time periods vastly different from posttreatment periods).
- Argument for Unit Fixed Effects:
  1. **Flexibility**: Increases model flexibility and thereby bolsters robustness.
  2. **Enhanced Precision**: Unit fixed effects explain a significant portion of outcome variation.
- SC Weighting & Unit Fixed Effects:
  - Under certain conditions, SC weighting can inherently account for unit fixed effects.
    - For example, when the weighted average outcome for control units in pretreatment is the same as that of the treated units. (unlikely in reality)
  - The use of unit fixed effect in synthetic control regression (i.e., synthetic control with intercept) was proposed before in Doudchenko and Imbens (2016) and Ferman and Pinto (2021) (called DIFP)

More details on application

1. Choose unit weights

- Regularization Parameter:
  - Equal to the size of a typical one-period outcome change for control units in the pre-period, then multiplied by a scaling factor (Arkhangelsky et al., 2021, p. 4092).
- Relation to SC Weights:
  - SDID weights are similar to those used in (Abadie et al., 2010) except two distinctions:
    1. Inclusion of an Intercept Term:
       * The weights in SynthDiD do not necessarily make the control pre-trends perfectly match the treated trends, just make them parallel.
       * This flexibility comes from the use of unit fixed effects, which can absorb any consistent differences between units.
    2. Regularization Penalty:
       * Adopted from Doudchenko and Imbens (2016) .
       * Enhances the dispersion and ensures the uniqueness of the weights.
  - DID weights are identical to those used in (Abadie et al., 2010) without intercept and regularization penalty and 1 treated unit.

2. Choose time weights

- Also include an intercept term, but no regularization (because correlated observations within time periods for the same unit is plausible, but not across units within the same period).

Note: To account for time-varying variables in the weights, one can use the residuals of the regression of the observed outcome on these time-varying variables, instead of the observed outcomes themselves ($Y_{it}^{res} = Y_{it} - X_{it}\hat{\beta}$, where $\hat{\beta}$ come from $Y = \beta X_{it}$).

The SDID method can account for systematic effects, often referred to as unit effects or unit heterogeneity, which influence treatment assignment (i.e., when treatment assignment is correlated with these systematic effects). Consequently, it provides unbiased estimates, especially valuable when there's a suspicion that the treatment might be influenced by persistent, unit-specific attributes.

Even in cases where we have completely random assignment, SDID, DiD, and SC are unbiased, but SynthDiD has the smallest SE.

28.2 Application

SDID Algorithm

1. Compute regularization parameter ζ

$$\zeta = (N_t T_{post})^{1/4} \hat{\sigma}$$

where

$$\hat{\sigma}^2 = \frac{1}{N_c(T_{pre}-1)} \sum_{i=1}^{N_c} \sum_{t=1}^{T_{re}-1} (\Delta_{it} - \hat{\Delta})^2$$

- $\Delta_{it} = Y_{i(t+1)} - Y_{it}$
- $\hat{\Delta} = \frac{1}{N_c(T_{pre}-1)} \sum_{i=1}^{N_c} \sum_{t=1}^{T_{pre}-1} \Delta_{it}$

2. Compute unit weights $\hat{w}^{sdid}$

$$(\hat{w}_0, \hat{w}^{sidid}) = \arg \min_{w_0 \in R, w \in \Omega} l_{unit}(w_0, w)$$

where

- $l_{unit}(w_0, w) = \sum_{t=1}^{T_{pre}} (w_0 + \sum_{i=1}^{N_c} w_i Y_{it} - \frac{1}{N_t} \sum_{i=N_c+1}^{N} Y_{it})^2 + \zeta^2 T_{pre} ||w||_2^2$
- $\Omega = \{w \in R_+^N : \sum_{i=1}^{N_c} w_i = 1, w_i = N_t^{-1} \forall i = N_c + 1, \dots, N\}$

3. Compute time weights $\hat{\lambda}^{sdid}$

$$(\hat{\lambda}_0, \hat{\lambda}^{sdid}) = \arg \min_{\lambda_0 \in R, \lambda \in \Lambda} l_{time}(\lambda_0, \lambda)$$

where

- $l_{time}(\lambda_0, \lambda) = \sum_{i=1}^{N_c} (\lambda_0 + \sum_{t=1}^{T_{pre}} \lambda_t Y_{it} - \frac{1}{T_{post}} \sum_{t=T_{pre}+1}^{T} Y_{it})^2$
- $\Lambda = \{\lambda \in R_+^T : \sum_{t=1}^{T_{pre}} \lambda_t = 1, \lambda_t = T_{post}^{-1} \forall t = T_{pre} + 1, \dots, T\}$

4. Compute the SDID estimator

$$(\hat{\tau}^{sdid}, \hat{\mu}, \hat{\alpha}, \hat{\beta}) = \arg \min_{\tau, \mu, \alpha, \beta} \{ \sum_{i=1}^{N} \sum_{t=1}^{T} (Y_{it} - \mu - \alpha_i - \beta_t - W_{it}\tau)^2 \hat{w}_i^{sdid} \hat{\lambda}_t^{sdid}$$

SE Estimation

- Under certain assumptions (errors, samples, and interaction properties between time and unit fixed effects) detailed in (Arkhangelsky et al., 2019, p. 4107), SDID is asymptotically normal and zero-centered
- Using its asymptotic variance, conventional confidence intervals can be applied to SDID.

$$\tau \in \hat{\tau}^{sdid} \pm z_{\alpha/2}\sqrt{\hat{V}_\tau}$$

- There are 3 approaches for variance estimation in confidence intervals:
  1. **Clustered Bootstrap (Efron, 1992):**
     - Independently resample units.
     - Advantages: Simple to use; robust performance in large panels due to natural approach to inference with panel data where observations of the same unit might be correlated.
     - Disadvantage: Computationally expensive.
  2. **Jackknife (Miller, 1974):**
     - Applied to weighted SDID regression with fixed weights.
     - Generally conservative and precise when treated and control units are sufficiently similar.
     - Not recommended for some methods, like the SC estimator, due to potential biases.
     - Appropriate for jackknifing DID without random weights.
  3. **Placebo Variance Estimation:**
     - Can used in cases with only one treated unit or large panels.
     - Placebo evaluations swap out the treated unit for untreated ones to estimate noise.
     - Relies on homoskedasticity across units.
     - Depends on homoskedasticity across units. It hinges on the empirical distribution of residuals from placebo estimators on control units.
     - The validity of the placebo method hinges on consistent noise distribution across units. One treated unit makes nonparametric variance estimation difficult, necessitating homoskedasticity for feasible inference. Detailed analysis available in Conley and Taber (2011).

All algorithms are from Arkhangelsky et al. (2021), p. 4109:

Bootstrap Variance Estimation

1. For each b from $1 \to B$:
   - Sample N rows from $(\mathbf{Y}, \mathbf{W})$ to get $(\mathbf{Y}^{(b)}, \mathbf{W}^{(b)})$ with replacement.
   - If the sample lacks treated or control units, resample.
   - Calculate $\tau^{(b)}$ using $(\mathbf{Y}^{(b)}, \mathbf{W}^{(b)})$.
2. Calculate variance: $\hat{V}_\tau = \frac{1}{B}\sum_{b=1}^{B}(\hat{\tau}^b - \frac{1}{B}\sum_{b=1}^{B}\hat{\tau}^b)^2$

Jackknife Variance Estimation

1. For each i from $1 \to N$:
   1. Calculate $\hat{\tau}^{(-i)}$: $\arg\min_{\tau,\{\alpha_j,\beta_t\}}\sum_{j\neq,i,t}(\mathbf{Y}_{jt} - \alpha_j - \beta_t - \tau\mathbf{W}_{it})^2\hat{w}_j\hat{\lambda}_t$
2. Calculate: $\hat{V}_\tau = (N-1)N^{-1}\sum_{i=1}^{N}(\hat{\tau}^{(-i)} - \hat{\tau})^2$

Placebo Variance Estimation

1. For each b from $1 \to B$
   1. Sample N_t out of N_c without replacement to get the "placebo" treatment
   2. Construct a placebo treatment matrix $\mathbf{W}_c^b$ for the controls
   3. Calculate $\hat{\tau}$ based on $(\mathbf{Y}_c, \mathbf{W}_c^b)$
2. Calculate $\hat{V}_\tau = \frac{1}{B}\sum_{b=1}^{B}(\hat{\tau}^b - \frac{1}{B}\sum_{b=1}^{B}\hat{\tau}^b)^2$

28.2.1 Block Treatment

Code provided by the `synthdid` package

```
library(synthdid)
library(tidyverse)

# Estimate the effect of California Proposition 99 on cigarette consumption
data('california_prop99')

setup = synthdid::panel.matrices(synthdid::california_prop99)

tau.hat = synthdid::synthdid_estimate(setup$Y, setup$N0, setup$T0)

# se = sqrt(vcov(tau.hat, method = 'placebo'))

plot(tau.hat) + causalverse::ama_theme()
```

```
setup = synthdid::panel.matrices(synthdid::california_prop99)

# Run for specific estimators
results_selected = causalverse::panel_estimate(setup,
                                               selected_estimators = c("synthdid", "dic

results_selected
#> $synthdid
#> $synthdid$estimate
#> synthdid: -15.604 +- NA. Effective N0/N0 = 16.4/38~0.4. Effective T0/T0 = 2.8/19~0.
#>
#> $synthdid$std.error
#> [1] 10.05324
#>
#>
#> $did
#> $did$estimate
#> synthdid: -27.349 +- NA. Effective N0/N0 = 38.0/38~1.0. Effective T0/T0 = 19.0/19~1
#>
#> $did$std.error
#> [1] 15.81479
#>
#>
#> $sc
#> $sc$estimate
#> synthdid: -19.620 +- NA. Effective N0/N0 = 3.8/38~0.1. Effective T0/T0 = Inf/19~Inf
#>
```

```
#> $sc$std.error
#> [1] 11.16422

# to access more details in the estimate object
summary(results_selected$did$estimate)
#> $estimate
#> [1] -27.34911
#>
#> $se
#>      [,1]
#> [1,]   NA
#>
#> $controls
#>                 estimate 1
#> Wyoming              0.026
#> Wisconsin            0.026
#> West Virginia        0.026
#> Virginia             0.026
#> Vermont              0.026
#> Utah                 0.026
#> Texas                0.026
#> Tennessee            0.026
#> South Dakota         0.026
#> South Carolina       0.026
#> Rhode Island         0.026
#> Pennsylvania         0.026
#> Oklahoma             0.026
#> Ohio                 0.026
#> North Dakota         0.026
#> North Carolina       0.026
#> New Mexico           0.026
#> New Hampshire        0.026
#> Nevada               0.026
#> Nebraska             0.026
#> Montana              0.026
#> Missouri             0.026
#> Mississippi          0.026
#> Minnesota            0.026
#> Maine                0.026
#> Louisiana            0.026
#> Kentucky             0.026
#> Kansas               0.026
#> Iowa                 0.026
#> Indiana              0.026
#> Illinois             0.026
```

```
#> Idaho                0.026
#> Georgia              0.026
#> Delaware             0.026
#> Connecticut          0.026
#>
#> $periods
#>      estimate 1
#> 1988      0.053
#> 1987      0.053
#> 1986      0.053
#> 1985      0.053
#> 1984      0.053
#> 1983      0.053
#> 1982      0.053
#> 1981      0.053
#> 1980      0.053
#> 1979      0.053
#> 1978      0.053
#> 1977      0.053
#> 1976      0.053
#> 1975      0.053
#> 1974      0.053
#> 1973      0.053
#> 1972      0.053
#> 1971      0.053
#>
#> $dimensions
#>           N1           N0 N0.effective           T1           T0 T0.effective
#>            1           38           38           12           19           19
```

```
causalverse::process_panel_estimate(results_selected)
#>     Method Estimate    SE
#> 1 SYNTHDID   -15.60 10.05
#> 2      DID   -27.35 15.81
#> 3       SC   -19.62 11.16
```

28.2.2 Staggered Adoption

To apply to staggered adoption settings using the SDID estimator (see examples in Arkhangelsky et al. (2021), p. 4115 similar to Ben-Michael et al. (2022)), we can:

1. Apply the SDID estimator repeatedly, once for every adoption date.
2. Using Ben-Michael et al. (2022) 's method, form matrices for each adoption date. Apply SDID and average based on treated unit/time-period

fractions.

3. Create multiple samples by splitting the data up by time periods. Each sample should have a consistent adoption date.

For a formal note on this special case, see Porreca (2022). It compares the outcomes from using SynthDiD with those from other estimators:

- Two-Way Fixed Effects (TWFE),
- The group time average treatment effect estimator from Callaway and Sant'Anna (2021),
- The partially pooled synthetic control method estimator from Ben-Michael et al. (2021), in a staggered treatment adoption context.
- The findings reveal that SynthDiD produces a different estimate of the average treatment effect compared to the other methods.
  - Simulation results suggest that these differences could be due to the SynthDiD's data generating process assumption (a latent factor model) aligning more closely with the actual data than the additive fixed effects model assumed by traditional DiD methods.

To explore heterogeneity of treatment effect, we can do subgroup analysis (Berman and Israeli, 2022, p. 1092)

Method	**Advantages**	**Disadvantages**	**Procedure**
Split Data into Subsets	Compares treated units to control units within the same subgroup.	Each subset uses a different synthetic control, making it challenging to compare effects across subgroups.	1. Split the data into separate subsets for each subgroup. 2. Compute synthetic DID effects for each subset.

Method	Advantages	Disadvantages	Procedure
Control Group Comprising All Non-adopters	Control weights match pretrends well for each treated subgroup.	Each control unit receives a different weight for each treatment subgroup, making it difficult to compare results due to varying synthetic controls.	1. Use a control group consisting of all non-adopters in each balanced panel cohort analysis. 2. Switch treatment units to the subgroup being analyzed. 3. Perform `synthdid` analysis.
Use All Data to Estimate Synthetic Control Weights **(recommend)**	All units have the same synthetic control.	Pretrend match may not be as accurate since it aims to match the average outcome of all treated units, not just a specific subgroup.	1. Use all the data to estimate the synthetic DID control weights. 2. Compute treatment effects using only the treated subgroup units as the treatment units.

```r
library(tidyverse)
df <- fixest::base_stagg |>
   dplyr::mutate(treatvar = if_else(time_to_treatment >= 0, 1, 0)) |>
   dplyr::mutate(treatvar = as.integer(if_else(year_treated > (5 + 2), 0, treatvar)))
```

```r
est <- causalverse::synthdid_est_ate(
  data               = df,
  adoption_cohorts   = 5:7,
  lags               = 2,
  leads              = 2,
  time_var           = "year",
  unit_id_var        = "id",
  treated_period_var = "year_treated",
  treat_stat_var     = "treatvar",
  outcome_var        = "y"
)
#> Adoption Cohort: 5
#> Treated units: 5 Control units: 65
#> Adoption Cohort: 6
#> Treated units: 5 Control units: 60
#> Adoption Cohort: 7
#> Treated units: 5 Control units: 55

data.frame(
    Period = names(est$TE_mean_w),
    ATE    = est$TE_mean_w,
    SE     = est$SE_mean_w
) |>
    causalverse::nice_tab()
#>    Period   ATE   SE
#> 1      -2 -0.05 0.22
#> 2      -1  0.05 0.22
#> 3       0 -5.07 0.80
#> 4       1 -4.68 0.51
#> 5       2 -3.70 0.79
#> 6 cumul.0 -5.07 0.80
#> 7 cumul.1 -4.87 0.55
#> 8 cumul.2 -4.48 0.53


causalverse::synthdid_plot_ate(est)
```

```
est_sub <- causalverse::synthdid_est_ate(
  data                = df,
  adoption_cohorts    = 5:7,
  lags                = 2,
  leads               = 2,
  time_var            = "year",
  unit_id_var         = "id",
  treated_period_var = "year_treated",
  treat_stat_var      = "treatvar",
  outcome_var         = "y",
  # a vector of subgroup id (from unit id)
  subgroup            =  c(
    # some are treated
    "11", "30", "49" ,
    # some are control within this period
    "20", "25", "21")
)
#> Adoption Cohort: 5
#> Treated units: 3 Control units: 65
#> Adoption Cohort: 6
#> Treated units: 0 Control units: 60
#> Adoption Cohort: 7
#> Treated units: 0 Control units: 55

data.frame(
    Period = names(est_sub$TE_mean_w),
    ATE = est_sub$TE_mean_w,
```

```
    SE = est_sub$SE_mean_w
) |>
    causalverse::nice_tab()
#>    Period   ATE   SE
#> 1      -2  0.32 0.44
#> 2      -1 -0.32 0.44
#> 3       0 -4.29 1.68
#> 4       1 -4.00 1.52
#> 5       2 -3.44 2.90
#> 6 cumul.0 -4.29 1.68
#> 7 cumul.1 -4.14 1.52
#> 8 cumul.2 -3.91 1.82

causalverse::synthdid_plot_ate(est)
```

Plot different estimators

```
library(causalverse)
methods <- c("synthdid", "did", "sc", "sc_ridge", "difp", "difp_ridge")

estimates <- lapply(methods, function(method) {
  synthdid_est_ate(
    data               = df,
    adoption_cohorts   = 5:7,
    lags               = 2,
    leads              = 2,
    time_var           = "year",
```

```
    unit_id_var        = "id",
    treated_period_var = "year_treated",
    treat_stat_var     = "treatvar",
    outcome_var        = "y",
    method = method
  )
})

plots <- lapply(seq_along(estimates), function(i) {
  causalverse::synthdid_plot_ate(estimates[[i]],
                                 title = methods[i],
                                 theme = causalverse::ama_theme(base_size = 6))
})

gridExtra::grid.arrange(grobs = plots, ncol = 2)
```

Chapter 29

Difference-in-differences

List of packages

Examples in marketing

- (Liaukonyte et al., 2015): TV ad on online shopping
- (Wang et al., 2018): political ad source and message tone on vote shares and turnout using discontinuities in the level of political ads at the borders
- (Datta et al., 2018): streaming service on total music consumption using timing of users adoption of a music streaming service
- (Janakiraman et al., 2018): data breach announcement affect customer spending using timing of data breach and variation whether customer info was breached in that event
- (Israeli, 2018): digital monitoring and enforcement on violations using enforcement of min ad price policies
- (Ramani and Srinivasan, 2019): firms respond to foreign direct investment liberalization using India's reform in 1991.
- (Pattabhiramaiah et al., 2019): paywall affects readership
- (Akca and Rao, 2020): aggregators for airlines business effect
- (Lim et al., 2020): nutritional labels on nutritional quality for other brands in a category using variation in timing of adoption of nutritional labels across categories
- (Guo et al., 2020): payment disclosure laws effect on physician prescription behavior using Timing of the Massachusetts open payment law as the exogenous shock
- (He et al., 2022): using Amazon policy change to examine the causal impact of fake reviews on sales, average ratings.
- (Peukert et al., 2022): using European General data protection Regulation, examine the impact of policy change on website usage.

Examples in econ:

- (Rosenzweig and Wolpin, 2000)
- (Angrist and Krueger, 2001)
- (Fuchs-Schündeln and Hassan, 2016): macro

Show the mechanism via

- Mediation Under DiD analysis: see (Habel et al., 2021)
- Moderation analysis: see (Goldfarb and Tucker, 2011)

Steps to trust DID:

1. Visualize the treatment rollout (e.g., `panelView`).
2. Document the number of treated units in each cohort (e.g., control and treated).
3. Visualize the trajectory of average outcomes across cohorts (if you have multiple periods).
4. Parallel Trends Conduct an event-study analysis with and without covariates.
5. For the case with covariates, check for overlap in covariates between treated and control groups to ensure control group validity (e.g., if the control is relatively small than the treated group, you might not have overlap, and you have to make extrapolation).
6. Conduct sensitivity analysis for parallel trend violations (e.g., `honestDiD`).

29.1 Visualization

```
library(panelView)
library(fixest)
library(tidyverse)
base_stagg <- fixest::base_stagg |>
    # treatment status
    dplyr::mutate(treat_stat = dplyr::if_else(time_to_treatment < 0, 0, 1)) |>
    select(id, year, treat_stat, y)

head(base_stagg)
#>   id year treat_stat           y
#> 2 90    1          0  0.01722971
#> 3 89    1          0 -4.58084528
#> 4 88    1          0  2.73817174
#> 5 87    1          0 -0.65103066
#> 6 86    1          0 -5.33381664
#> 7 85    1          0  0.49562631
```

```r
panelView::panelview(
    y ~ treat_stat,
    data = base_stagg,
    index = c("id", "year"),
    xlab = "Year",
    ylab = "Unit",
    display.all = F,
    gridOff = T,
    by.timing = T
)
```

```r
# alternatively specification
panelView::panelview(
    Y = "y",
    D = "treat_stat",
    data = base_stagg,
    index = c("id", "year"),
    xlab = "Year",
    ylab = "Unit",
    display.all = F,
    gridOff = T,
    by.timing = T
)
```

```r
# Average outcomes for each cohort
panelView::panelview(
    data = base_stagg,
    Y = "y",
    D = "treat_stat",
    index = c("id", "year"),
    by.timing = T,
    display.all = F,
    type = "outcome",
    by.cohort = T
)
#> Number of unique treatment histories: 10
```

29.2 Simple Dif-n-dif

- A tool developed intuitively to study "natural experiment", but its uses are much broader.
- Fixed Effects Estimator is the foundation for DID
- Why is dif-in-dif attractive? Identification strategy: Inter-temporal variation between groups
  - **Cross-sectional estimator** helps avoid omitted (unobserved) **common trends**
  - **Time-series estimator** helps overcome omitted (unobserved) **cross-sectional differences**

Consider

- $D_i = 1$ treatment group
- $D_i = 0$ control group
- $T = 1$ After the treatment
- $T = 0$ Before the treatment

	After (T = 1)	Before (T = 0)
Treated $D_i = 1$	$E[Y_{1i}(1)\|D_i = 1]$	$E[Y_{0i}(0)\|D)i = 1]$
Control $D_i = 0$	$E[Y_{0i}(1)\|D_i = 0]$	$E[Y_{0i}(0)\|D_i = 0]$

missing $E[Y_{0i}(1)|D=1]$

The Average Treatment Effect on Treated (ATT)

$$\begin{aligned} E[Y_1(1) - Y_0(1)|D=1] = &\{E[Y(1)|D=1] - E[Y(1)|D=0]\} \\ &- \{E[Y(0)|D=1] - E[Y(0)|D=0]\} \end{aligned}$$

More elaboration:

- For the treatment group, we isolate the difference between being treated and not being treated. If the untreated group would have been affected in a different way, the DiD design and estimate would tell us nothing.
- Alternatively, because we can't observe treatment variation in the control group, we can't say anything about the treatment effect on this group.

Extension

1. **More than 2 groups** (multiple treatments and multiple controls), and more than 2 period (pre and post)

$$Y_{igt} = \alpha_g + \gamma_t + \beta I_{gt} + \delta X_{igt} + \epsilon_{igt}$$

where

- α_g is the group-specific fixed effect
- γ_t = time specific fixed effect
- β = dif-in-dif effect
- I_{gt} = interaction terms (n treatment indicators x n post-treatment dummies) (capture effect heterogeneity over time)

This specification is the "two-way fixed effects DiD" - **TWFE** (i.e., 2 sets of fixed effects: group + time).

- However, if you have Staggered Dif-n-dif (i.e., treatment is applied at different times to different groups). TWFE is really bad.

2. **Long-term Effects**

To examine the dynamic treatment effects (that are not under rollout/staggered design), we can create a centered time variable,

Centered Time Variable	Period
...	
$t = -1$	2 periods before treatment period
$t = 0$	Last period right before treatment period Remember to use this period as reference group
$t = 1$	Treatment period

Centered Time Variable	Period
…	

By interacting this factor variable, we can examine the dynamic effect of treatment (i.e., whether it's fading or intensifying)

$$
\begin{aligned}
Y &= \alpha_0 + \alpha_1 Group + \alpha_2 Time \\
&+ \beta_{-T_1} Treatment + \beta_{-(T_1-1)} Treatment + \cdots + \beta_{-1} Treatment \\
&+ \beta_1 + \cdots + \beta_{T_2} Treatment
\end{aligned}
$$

where

- β_0 is used as the reference group (i.e., drop from the model)
- T_1 is the pre-treatment period
- T_2 is the post-treatment period

With more variables (i.e., interaction terms), coefficients estimates can be less precise (i.e., higher SE).

3. DiD on the relationship, not levels. Technically, we can apply DiD research design not only on variables, but also on coefficients estimates of some other regression models with before and after a policy is implemented.

Goal:

1. Pre-treatment coefficients should be non-significant $\beta_{-T_1}, \ldots, \beta_{-1} = 0$ (similar to the Placebo Test)
2. Post-treatment coefficients are expected to be significant $\beta_1, \ldots, \beta_{T_2} \neq 0$
    - You can now examine the trend in post-treatment coefficients (i.e., increasing or decreasing)

```
library(tidyverse)
library(fixest)

od <- causaldata::organ_donations %>%

    # Treatment variable
    dplyr::mutate(California = State == 'California') %>%
    # centered time variable
    dplyr::mutate(center_time = as.factor(Quarter_Num - 3))
# where 3 is the reference period precedes the treatment period

class(od$California)
#> [1] "logical"
class(od$State)
```

```
#> [1] "character"

cali <- feols(Rate ~ i(center_time, California, ref = 0) |
                  State + center_time,
              data = od)

etable(cali)
#>                                              cali
#> Dependent Var.:                              Rate
#>
#> California x center_time = -2    -0.0029 (0.0051)
#> California x center_time = -1   0.0063** (0.0023)
#> California x center_time = 1  -0.0216*** (0.0050)
#> California x center_time = 2  -0.0203*** (0.0045)
#> California x center_time = 3    -0.0222* (0.0100)
#> Fixed-Effects:                 -------------------
#> State                                         Yes
#> center_time                                   Yes
#> ____________________________ ___________________
#> S.E.: Clustered                           by: State
#> Observations                                    162
#> R2                                          0.97934
#> Within R2                                   0.00979
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

iplot(cali, pt.join = T)
```

```
coefplot(cali)
```

29.3 Notes

- Matching Methods
  - Match treatment and control based on pre-treatment observables
  - Modify SEs appropriately (Heckman et al., 1997). It's might be eas-

ier to just use the Doubly Robust DiD (Sant'Anna and Zhao, 2020) where you just need either matching or regression to work in order to identify your treatment effect

  - Whereas the group fixed effects control for the group time-invariant effects, it does not control for selection bias (i.e., certain groups are more likely to be treated than others). Hence, with these backdoor open (i.e., selection bias) between (1) propensity to be treated and (2) dynamics evolution of the outcome post-treatment, matching can potential close these backdoor.
  - Be careful when matching time-varying covariates because you might encounter "regression to the mean" problem, where pre-treatment periods can have an unusually bad or good time (that is out of the ordinary), then the post-treatment period outcome can just be an artifact of the regression to the mean (Daw and Hatfield, 2018). This problem is not of concern to time-invariant variables.
  - Matching and DiD can use pre-treatment outcomes to correct for selection bias. From real world data and simulation, (Chabé-Ferret, 2015) found that matching generally underestimates the average causal effect and gets closer to the true effect with more number of pre-treatment outcomes. When selection bias is symmetric around the treatment date, DID is still consistent when implemented symmetrically (i.e., the same number of period before and after treatment). In cases where selection bias is asymmetric, the MC simulations show that Symmetric DiD still performs better than Matching.
  - Forward DID is a simple algo that helps choose control units (Li, 2024).

- It's always good to show results with and without controls because
  - If the controls are fixed within group or within time, then those should be absorbed under those fixed effects
  - If the controls are dynamic across group and across, then your parallel trends assumption is not plausible.
- Under causal inference, R^2 is not so important.

For count data, one can use the fixed-effects Poisson pseudo-maximum likelihood estimator (PPML) (Athey and Imbens, 2006, Puhani (2012)) (For applied papers, see Burtch et al. (2018) in management and He et al. (2021) in marketing). This also allows for robust standard errors under over-dispersion (Wooldridge, 1999).

- This estimator outperforms a log OLS when data have many 0s(Silva and Tenreyro, 2011), since log-OLS can produce biased estimates (O'Hara and Kotze, 2010) under heteroskedascity (Silva and Tenreyro, 2006).

- For those thinking of negative binomial with fixed effects, there isn't an estimator right now (Allison and Waterman, 2002).

For [Zero-valued Outcomes], we have to distinguish the treatment effect on the intensive (outcome: 10 to 11) vs. extensive margins (outcome: 0 to 1), and we can't readily interpret the treatment coefficient of log-transformed outcome regression as percentage change (Chen and Roth, 2023). Alternatively, we can either focus on

- **Proportional treatment effects**: $\theta_{ATT\%} = \frac{E(Y_{it}(1)|D_i=1,Post_t=1)-E(Y_{it}(0)|D_i=1,Post_t=1)}{E(Y_{it}(0)|D_i=1,Post_t=1}$ (i.e., percentage change in treated group's average post-treatment outcome). Instead of relying on the parallel trends assumption in levels, we could also rely on parallel trends assumption in ratio (Wooldridge, 2023).
  - We can use Poisson QMLE to estimate the treatment effect: $Y_{it} = \exp(\beta_0 + D_i \times \beta_1 Post_t + \beta_2 D_i + \beta_3 Post_t + X_{it})\epsilon_{it}$ and $\hat{\theta}_{ATT\%} = \exp(\hat{\beta}_1 - 1)$.
  - To examine the parallel trends assumption in ratio holds, we can also estimate a dynamic version of the Poisson QMLE: $Y_{it} = \exp(\lambda_t + \beta_2 D_i + \sum_{r \neq -1} \beta_r D_i \times (RelativeTime_t = r)$, we would expect $\exp(\hat{\beta}_r) - 1 = 0$ for $r < 0$.
  - Even if we see the plot of these coefficients are 0, we still should run sensitivity analysis (Rambachan and Roth, 2023) to examine violation of this assumption (see Prior Parallel Trends Test).
- **Log Effects with Calibrated Extensive-margin value**: due to problem with the mean value interpretation of the proportional treatment effects with outcomes that are heavy-tailed, we might be interested in the extensive margin effect. Then, we can explicit model how much weight we put on the intensive vs. extensive margin (Chen and Roth, 2023, p. 39).

1. **Proportional treatment effects**

```r
set.seed(123) # For reproducibility

n <- 500 # Number of observations per group (treated and control)
# Generating IDs for a panel setup
ID <- rep(1:n, times = 2)

# Defining groups and periods
Group <- rep(c("Control", "Treated"), each = n)
Time <- rep(c("Before", "After"), times = n)
Treatment <- ifelse(Group == "Treated", 1, 0)
Post <- ifelse(Time == "After", 1, 0)

# Step 1: Generate baseline outcomes with a zero-inflated model
lambda <- 20 # Average rate of occurrence
```

```r
zero_inflation <- 0.5 # Proportion of zeros
Y_baseline <-
    ifelse(runif(2 * n) < zero_inflation, 0, rpois(2 * n, lambda))

# Step 2: Apply DiD treatment effect on the treated group in the post-treatment period
Treatment_Effect <- Treatment * Post
Y_treatment <-
    ifelse(Treatment_Effect == 1, rpois(n, lambda = 2), 0)

# Incorporating a simple time trend, ensuring outcomes are non-negative
Time_Trend <- ifelse(Time == "After", rpois(2 * n, lambda = 1), 0)

# Step 3: Combine to get the observed outcomes
Y_observed <- Y_baseline + Y_treatment + Time_Trend

# Ensure no negative outcomes after the time trend
Y_observed <- ifelse(Y_observed < 0, 0, Y_observed)

# Create the final dataset
data <-
    data.frame(
        ID = ID,
        Treatment = Treatment,
        Period = Post,
        Outcome = Y_observed
    )

# Viewing the first few rows of the dataset
head(data)
#>   ID Treatment Period Outcome
#> 1  1         0      0       0
#> 2  2         0      1      25
#> 3  3         0      0       0
#> 4  4         0      1      20
#> 5  5         0      0      19
#> 6  6         0      1       0
```

```r
library(fixest)
res_pois <-
    fepois(Outcome ~ Treatment + Period + Treatment * Period,
           data = data,
           vcov = "hetero")
etable(res_pois)
#>                              res_pois
#> Dependent Var.:               Outcome
```

```
#>
#> Constant           2.249*** (0.0717)
#> Treatment           0.1743. (0.0932)
#> Period               0.0662 (0.0960)
#> Treatment x Period   0.0314 (0.1249)
#> ------------------ -----------------
#> S.E. type          Heteroskeda.-rob.
#> Observations                   1,000
#> Squared Cor.                 0.01148
#> Pseudo R2                    0.00746
#> BIC                        15,636.8
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Average percentage change
exp(coefficients(res_pois)["Treatment:Period"]) - 1
#> Treatment:Period
#>       0.03191643

# SE using delta method
exp(coefficients(res_pois)["Treatment:Period"]) *
    sqrt(res_pois$cov.scaled["Treatment:Period", "Treatment:Period"])
#> Treatment:Period
#>        0.1288596
```

In this example, the DID coefficient is not significant. However, say that it's significant, we can interpret the coefficient as 3 percent increase in posttreatment period due to the treatment.

```
library(fixest)

base_did_log0 <- base_did |>
    mutate(y = if_else(y > 0, y, 0))

res_pois_es <-
    fepois(y ~ x1 + i(period, treat, 5) | id + period,
           data = base_did_log0,
           vcov = "hetero")

etable(res_pois_es)
#>                            res_pois_es
#> Dependent Var.:                      y
#>
#> x1                  0.1895*** (0.0108)
#> treat x period = 1    -0.2769 (0.3545)
#> treat x period = 2    -0.2699 (0.3533)
```

```
#> treat x period = 3     0.1737 (0.3520)
#> treat x period = 4    -0.2381 (0.3249)
#> treat x period = 6     0.3724 (0.3086)
#> treat x period = 7    0.7739* (0.3117)
#> treat x period = 8    0.5028. (0.2962)
#> treat x period = 9   0.9746** (0.3092)
#> treat x period = 10  1.310*** (0.3193)
#> Fixed-Effects:      ------------------
#> id                                Yes
#> period                            Yes
#> ------------------- ------------------
#> S.E. type           Heteroskedas.-rob.
#> Observations                    1,080
#> Squared Cor.                  0.51131
#> Pseudo R2                     0.34836
#> BIC                           5,868.8
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
iplot(res_pois_es)
```

This parallel trend is the "ratio" version as in Wooldridge (2023) :

$$\frac{E(Y_{it}(0)|D_i = 1, Post_t = 1)}{E(Y_{it}(0)|D_i = 1, Post_t = 0)} = \frac{E(Y_{it}(0)|D_i = 0, Post_t = 1)}{E(Y_{it}(0)|D_i = 0, Post_t = 0)}$$

which means without treatment, the average percentage change in the mean outcome for treated group is identical to that of the control group.

2. **Log Effects with Calibrated Extensive-margin value**

If we want to study the treatment effect on a concave transformation of the outcome that is less influenced by those in the distribution's tail, then we can perform this analysis.

Steps:

1. Normalize the outcomes such that 1 represents the minimum non-zero and positve value (i.e., divide the outcome by its minimum non-zero and positive value).
2. Estimate the treatment effects for the new outcome

$$m(y) = \begin{cases} \log(y) & \text{for } y > 0 \\ -x & \text{for } y = 0 \end{cases}$$

The choice of x depends on what the researcher is interested in:

Value of x	Interest
$x = 0$	The treatment effect in logs where all zero-valued outcomes are set to equal the minimum non-zero value (i.e., we exclude the extensive-margin change between 0 and y_{min})
$x > 0$	Setting the change between 0 and y_{min} to be valued as the equivalent of a x log point change along the intensive margin.

```
library(fixest)
base_did_log0_cali <- base_did_log0 |>
    # get min
    mutate(min_y = min(y[y > 0])) |>

    # normalized the outcome
    mutate(y_norm = y / min_y)

my_regression <-
    function(x) {
        base_did_log0_cali <-
            base_did_log0_cali %>% mutate(my = ifelse(y_norm == 0,-x,
                                                      log(y_norm)))
        my_reg <-
            feols(
                fml = my ~ x1 + i(period, treat, 5) | id + period,
                data = base_did_log0_cali,
                vcov = "hetero"
            )
```

```
        return(my_reg)
    }

xvec <- c(0, .1, .5, 1, 3)
reg_list <- purrr::map(.x = xvec, .f = my_regression)


iplot(reg_list,
      pt.col =  1:length(xvec),
      pt.pch = 1:length(xvec))
legend("topleft",
       col = 1:length(xvec),
       pch = 1:length(xvec),
       legend = as.character(xvec))
```

```
etable(
    reg_list,
    headers = list("Extensive-margin value (x)" = as.character(xvec)),
    digits = 2,
    digits.stats = 2
)
#>                                   model 1        model 2        model 3
#> Extensive-margin value (x)              0            0.1            0.5
#> Dependent Var.:                        my             my             my
#>
```

```
#> x1                          0.43*** (0.02) 0.44*** (0.02) 0.46*** (0.03)
#> treat x period = 1            -0.92 (0.67)   -0.94 (0.69)    -1.0 (0.73)
#> treat x period = 2            -0.41 (0.66)   -0.42 (0.67)   -0.43 (0.71)
#> treat x period = 3            -0.34 (0.67)   -0.35 (0.68)   -0.38 (0.73)
#> treat x period = 4             -1.0 (0.67)    -1.0 (0.68)    -1.1 (0.73)
#> treat x period = 6             0.44 (0.66)    0.44 (0.67)    0.45 (0.72)
#> treat x period = 7             1.1. (0.64)    1.1. (0.65)    1.2. (0.70)
#> treat x period = 8             1.1. (0.64)    1.1. (0.65)     1.1 (0.69)
#> treat x period = 9            1.7** (0.65)   1.7** (0.66)    1.8* (0.70)
#> treat x period = 10          2.4*** (0.62)  2.4*** (0.63)  2.5*** (0.68)
#> Fixed-Effects:              -------------- -------------- --------------
#> id                                     Yes            Yes            Yes
#> period                                 Yes            Yes            Yes
#> -------------------------- -------------- -------------- --------------
#> S.E. type                   Heterosk.-rob. Heterosk.-rob. Heterosk.-rob.
#> Observations                         1,080          1,080          1,080
#> R2                                    0.43           0.43           0.43
#> Within R2                             0.26           0.26           0.25
#>
#>                                    model 4        model 5
#> Extensive-margin value (x)               1              3
#> Dependent Var.:                         my             my
#>
#> x1                          0.49*** (0.03) 0.62*** (0.04)
#> treat x period = 1             -1.1 (0.79)     -1.5 (1.0)
#> treat x period = 2            -0.44 (0.77)   -0.51 (0.99)
#> treat x period = 3            -0.43 (0.78)    -0.60 (1.0)
#> treat x period = 4             -1.2 (0.78)     -1.5 (1.0)
#> treat x period = 6             0.45 (0.77)     0.46 (1.0)
#> treat x period = 7              1.2 (0.75)     1.3 (0.97)
#> treat x period = 8              1.2 (0.74)     1.3 (0.96)
#> treat x period = 9             1.8* (0.75)    2.1* (0.97)
#> treat x period = 10          2.7*** (0.73)  3.2*** (0.94)
#> Fixed-Effects:              -------------- --------------
#> id                                     Yes            Yes
#> period                                 Yes            Yes
#> -------------------------- -------------- --------------
#> S.E. type                   Heterosk.-rob. Heterosk.-rob.
#> Observations                         1,080          1,080
#> R2                                    0.42           0.41
#> Within R2                             0.25           0.24
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We have the dynamic treatment effects for different hypothesized extensive-

margin value of $x \in (0, .1, .5, 1, 3, 5)$

The first column is when the zero-valued outcome equal to $y_{min,y>0}$ (i.e., there is no different between the minimum outcome and zero outcome - $x = 0$)

For this particular example, as the extensive margin increases, we see an increase in the effect magnitude. The second column is when we assume an extensive-margin change from 0 to $y_{min,y>0}$ is equivalent to a 10 (i.e., 0.1×100) log point change along the intensive margin.

29.4 Standard Errors

Serial correlation is a big problem in DiD because (Bertrand et al., 2004)

1. DiD often uses long time series
2. Outcomes are often highly positively serially correlated
3. Minimal variation in the treatment variable over time within a group (e.g., state).

To overcome this problem:

- Using parametric correction (standard AR correction) is not good.
- Using nonparametric (e.g., **block bootstrap**- keep all obs from the same group such as state together) is good when number of groups is large.
- Remove time series dimension (i.e., aggregate data into 2 periods: pre and post). This still works with small number of groups (See (Donald and Lang, 2007) for more notes on small-sample aggregation).
- Empirical and arbitrary variance-covariance matrix corrections work only in large samples.

29.5 Examples

Example by Philipp Leppert replicating Card and Krueger (1994)

Example by Anthony Schmidt

29.5.1 Example by Doleac and Hansen (2020)

- The purpose of banning a checking box for ex-criminal was banned because we thought that it gives more access to felons
- Even if we ban the box, employers wouldn't just change their behaviors. But then the unintended consequence is that employers statistically discriminate based on race

3 types of ban the box

1. Public employer only
2. Private employer with government contract

3. All employers

Main identification strategy

- If any county in the Metropolitan Statistical Area (MSA) adopts ban the box, it means the whole MSA is treated. Or if the state adopts "ban the ban," every county is treated

Under Simple Dif-n-dif

$$Y_{it} = \beta_0 + \beta_1 Post_t + \beta_2 treat_i + \beta_2(Post_t \times Treat_i) + \epsilon_{it}$$

But if there is no common post time, then we should use Staggered Dif-n-dif

$$\begin{aligned} E_{imrt} = \alpha &+ \beta_1 BTB_{imt} W_{imt} + \beta_2 BTB_{mt} + \beta_3 BTB_{mt} H_{imt} \\ &+ \delta_m + D_{imt}\beta_5 + \lambda_{rt} + \delta_m \times f(t)\beta_7 + e_{imrt} \end{aligned}$$

where

- i = person; m = MSA; r = region (US regions e.g., Midwest) ; r = region; t = year
- W = White; B = Black; H = Hispanic
- $\beta_1 BTB_{imt} W_{imt} + \beta_2 BTB_{mt} + \beta_3 BTB_{mt} H_{imt}$ are the 3 dif-n-dif variables (BTB = "ban the box")
- δ_m = dummy for MSI
- D_{imt} = control for people
- λ_{rt} = region by time fixed effect
- $\delta_m \times f(t)$ = linear time trend within MSA (but we should not need this if we have good pre-trend)

If we put $\lambda_r - \lambda_t$ (separately) we will more broad fixed effect, while λ_{rt} will give us deeper and narrower fixed effect.

Before running this model, we have to drop all other races. And $\beta_1, \beta_2, \beta_3$ are not collinear because there are all interaction terms with BTB_{mt}

If we just want to estimate the model for black men, we will modify it to be

$$E_{imrt} = \alpha + BTB_{mt}\beta_1 + \delta_m + D_{imt}\beta_5 + \lambda_{rt} + (\delta_m \times f(t))\beta_7 + e_{imrt}$$

$$\begin{aligned} E_{imrt} = \alpha &+ BTB_{m(t-3t)}\theta_1 + BTB_{m(t-2)}\theta_2 + BTB_{mt}\theta_4 \\ &+ BTB_{m(t+1)}\theta_5 + BTB_{m(t+2)}\theta_6 + BTB_{m(t+3t)}\theta_7 \\ &+ [\delta_m + D_{imt}\beta_5 + \lambda_r + (\delta_m \times (f(t))\beta_7 + e_{imrt}] \end{aligned}$$

We have to leave $BTB_{m(t-1)}\theta_3$ out for the category would not be perfect collinearity

So the year before BTB $(\theta_1, \theta_2, \theta_3)$ should be similar to each other (i.e., same pre-trend). Remember, we only run for places with BTB.

If θ_2 is statistically different from θ_3 (baseline), then there could be a problem, but it could also make sense if we have pre-trend announcement.

29.5.2 Example from Princeton

```
library(foreign)
mydata = read.dta("http://dss.princeton.edu/training/Panel101.dta") %>%
    # create a dummy variable to indicate the time when the treatment started
    dplyr::mutate(time = ifelse(year >= 1994, 1, 0)) %>%
    # create a dummy variable to identify the treatment group
    dplyr::mutate(treated = ifelse(country == "E" |
                                       country == "F" | country == "G" ,
                                   1,
                                   0)) %>%
    # create an interaction between time and treated
    dplyr::mutate(did = time * treated)
```

estimate the DID estimator

```
didreg = lm(y ~ treated + time + did, data = mydata)
summary(didreg)
#>
#> Call:
#> lm(formula = y ~ treated + time + did, data = mydata)
#>
#> Residuals:
#>        Min         1Q     Median         3Q        Max
#> -9.768e+09 -1.623e+09  1.167e+08  1.393e+09  6.807e+09
#>
#> Coefficients:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  3.581e+08  7.382e+08   0.485   0.6292
#> treated      1.776e+09  1.128e+09   1.575   0.1200
#> time         2.289e+09  9.530e+08   2.402   0.0191 *
#> did         -2.520e+09  1.456e+09  -1.731   0.0882 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 2.953e+09 on 66 degrees of freedom
#> Multiple R-squared:  0.08273,    Adjusted R-squared:  0.04104
#> F-statistic: 1.984 on 3 and 66 DF,  p-value: 0.1249
```

The `did` coefficient is the differences-in-differences estimator. Treat has a negative effect

29.5.3 Example by Card and Krueger (1993)

found that increase in minimum wage increases employment

Experimental Setting:

- New Jersey (treatment) increased minimum wage
- Penn (control) did not increase minimum wage

		After	Before	
Treatment	NJ	A	B	A - B
Control	PA	C	D	C - D
		A - C	B - D	(A - B) - (C - D)

where

- A - B = treatment effect + effect of time (additive)
- C - D = effect of time
- (A - B) - (C - D) = dif-n-dif

The identifying assumptions:

- Can't have **switchers**
- PA is the control group
  - is a good counter factual
  - is what NJ would look like if they hadn't had the treatment

$$Y_{jt} = \beta_0 + NJ_j\beta_1 + POST_t\beta_2 + (NJ_j \times POST_t)\beta_3 + X_{jt}\beta_4 + \epsilon_{jt}$$

where

- j = restaurant
- NJ = dummy where $1 = NJ$, and $0 = PA$
- $POST$ = dummy where $1 = post$, and $0 = pre$

Notes:

- We don't need β_4 in our model to have unbiased β_3, but including it would give our coefficients efficiency
- If we use ΔY_{jt} as the dependent variable, we don't need $POST_t\beta_2$ anymore

- Alternative model specification is that the authors use NJ high wage restaurant as control group (still choose those that are close to the border)
- The reason why they can't control for everything (PA + NJ high wage) is because it's hard to interpret the causal treatment
- Dif-n-dif utilizes similarity in pretrend of the dependent variables. However, this is neither a necessary nor sufficient for the identifying assumption.
  - It's not sufficient because they can have multiple treatments (technically, you could include more control, but your treatment can't interact)
  - It's not necessary because trends can be parallel after treatment
- However, we can't never be certain; we just try to find evidence consistent with our theory so that dif-n-dif can work.
- Notice that we don't need before treatment the **levels of the dependent variable** to be the same (e.g., same wage average in both NJ and PA), dif-n-dif only needs **pre-trend (i.e., slope)** to be the same for the two groups.

29.5.4 Example by Butcher et al. (2014)

Theory:

- Highest achieving students are usually in hard science. Why?
  - Hard to give students students the benefit of doubt for hard science
  - How unpleasant and how easy to get a job. Degrees with lower market value typically want to make you feel more pleasant

Under OLS

$$E_{ij} = \beta_0 + X_i\beta_1 + G_j\beta_2 + \epsilon_{ij}$$

where

- X_i = student attributes
- β_2 = causal estimate (from grade change)
- E_{ij} = Did you choose to enroll in major j
- G_j = grade given in major j

Examine $\hat{\beta}_2$

- Negative bias: Endogenous response because department with lower enrollment rate will give better grade

- Positive bias: hard science is already having best students (i.e., ability), so if they don't their grades can be even lower

Under dif-n-dif

$$Y_{idt} = \beta_0 + POST_t\beta_1 + Treat_d\beta_2 + (POST_t \times Treat_d)\beta_3 + X_{idt} + \epsilon_{idt}$$

where

- Y_{idt} = grade average

	Intercept	Treat	Post	Treat*Post
Treat Pre	1	1	0	0
Treat Post	1	1	1	1
Control Pre	1	0	0	0
Control Post	1	0	1	0
	Average for pre-control β_0			

A more general specification of the dif-n-dif is that

$$Y_{idt} = \alpha_0 + (POST_t \times Treat_d)\alpha_1 + \theta_d + \delta_t + X_{idt} + u_{idt}$$

where

- $(\theta_d + \delta_t)$ richer , more df than $Treat_d\beta_2 + Post_t\beta_1$ (because fixed effects subsume Post and treat)
- α_1 should be equivalent to β_3 (if your model assumptions are correct)

29.6 One Difference

The regression formula is as follows (Liaukonytė et al., 2023):

$$y_{ut} = \beta \text{Post}_t + \gamma_u + \gamma_w(t) + \gamma_l + \gamma_g(u)p(t) + \epsilon_{ut}$$

where

- y_{ut}: Outcome of interest for unit u in time t.
- Post_t: Dummy variable representing a specific post-event period.
- β: Coefficient measuring the average change in the outcome after the event relative to the pre-period.

- γ_u: Fixed effects for each unit.
- $\gamma_w(t)$: Time-specific fixed effects to account for periodic variations.
- γ_l: Dummy variable for a specific significant period (e.g., a major event change).
- $\gamma_g(u)p(t)$: Group x period fixed effects for flexible trends that may vary across different categories (e.g., geographical regions) and periods.
- ϵ_{ut}: Error term.

This model can be used to analyze the impact of an event on the outcome of interest while controlling for various fixed effects and time-specific variations, but using units themselves pre-treatment as controls.

29.7 Two-way Fixed-effects

A generalization of the dif-n-dif model is the two-way fixed-effects models where you have multiple groups and time effects. But this is not a designed-based, non-parametric causal estimator (Imai and Kim, 2021)

When applying TWFE to multiple groups and multiple periods, the supposedly causal coefficient is the weighted average of all two-group/two-period DiD estimators in the data where some of the weights can be negative. More specifically, the weights are proportional to group sizes and treatment indicator's variation in each pair, where units in the middle of the panel have the highest weight.

The canonical/standard TWFE only works when

- Effects are homogeneous across units and across time periods (i.e., no dynamic changes in the effects of treatment). See (Goodman-Bacon, 2021; De Chaisemartin and d'Haultfoeuille, 2020; Sun and Abraham, 2021; Borusyak et al., 2021) for details. Similarly, it relies on the assumption of **linear additive effects** (Imai and Kim, 2021)
  - Have to argue why treatment heterogeneity is not a problem (e.g., plot treatment timing and decompose treatment coefficient using Goodman-Bacon Decomposition) know the percentage of observation are never treated (because as the never-treated group increases, the bias of TWFE decreases, with 80% sample to be never-treated, bias is negligible). The problem is worsen when you have long-run effects.
  - Need to manually drop two relative time periods if everyone is eventually treated (to avoid multicollinearity). Programs might do this randomly and if it chooses to drop a post-treatment period, it will create biases. The choice usually -1, and -2 periods.
  - Treatment heterogeneity can come in because (1) it might take some time for a treatment to have measurable changes in outcomes or (2) for each period after treatment, the effect can be different (phase in or increasing effects).

- 2 time periods.

Within this setting, TWFE works because, using the baseline (e.g., control units where their treatment status is unchanged across time periods), the comparison can be

- Good for
  - Newly treated units vs. control
  - Newly treated units vs not-yet treated
- Bad for
  - Newly treated vs. already treated (because already treated cannot serve as the potential outcome for the newly treated).
  - Strict exogeneity (i.e., time-varying confounders, feedback from past outcome to treatment) (Imai and Kim, 2019)
  - Specific functional forms (i.e., treatment effect homogeneity and no carryover effects or anticipation effects) (Imai and Kim, 2019)

Note: Notation for this section is consistent with (ark, 2020)

$$Y_{it} = \alpha_i + \lambda_t + \tau W_{it} + \beta X_{it} + \epsilon_{it}$$

where

- Y_{it} is the outcome
- α_i is the unit FE
- λ_t is the time FE
- τ is the causal effect of treatment
- W_{it} is the treatment indicator
- X_{it} are covariates

When $T = 2$, the TWFE is the traditional DiD model

Under the following assumption, $\hat{\tau}_{OLS}$ is unbiased:

1. homogeneous treatment effect
2. parallel trends assumptions
3. linear additive effects (Imai and Kim, 2021)

Remedies for TWFE's shortcomings

- (Goodman-Bacon, 2021): diagnostic robustness tests of the TWFE DiD and identify influential observations to the DiD estimate (Goodman-Bacon Decomposition)
- (Callaway and Sant'Anna, 2021): 2-step estimation with a bootstrap procedure that can account for autocorrelation and clustering,

  - the parameters of interest are the group-time average treatment effects, where each group is defined by when it was first treated (Multiple periods and variation in treatment timing)
  - Comparing post-treatment outcomes fo groups treated in a period against a similar group that is never treated (using matching).
  - Treatment status cannot switch (once treated, stay treated for the rest of the panel)
  - Package: `did`
- (Sun and Abraham, 2021): a specialization of (Callaway and Sant'Anna, 2021) in the event-study context.
  - They include lags and leads in their design
  - have cohort-specific estimates (similar to group-time estimates in (Callaway and Sant'Anna, 2021)
  - They propose the "interaction-weighted" estimator.
  - Package: `fixest`
- (Imai and Kim, 2021)
  - Different from (Callaway and Sant'Anna, 2021) because they allow units to switch in and out of treatment.
  - Based on matching methods, to have weighted TWFE
  - Package: `wfe` and `PanelMatch`
- (Gardner, 2022): two-stage DiD
  - `did2s`
- In cases with an unaffected unit (i.e., never-treated), using the exposure-adjusted difference-in-differences estimators can recover the average treatment effect (De Chaisemartin and d'Haultfoeuille, 2020). However, if you want to see the treatment effect heterogeneity (in cases where the true heterogeneous treatment effects vary by the exposure rate), exposure-adjusted did still fails (Sun and Shapiro, 2022).
- (ark, 2020): see below

To be robust against

1. time- and unit-varying effects

We can use the reshaped inverse probability weighting (RIPW)- TWFE estimator

With the following assumptions:

- SUTVA

- Binary treatment: $\mathbf{W}_i = (W_{i1}, \dots, W_{it})$ where $\mathbf{W}_i \sim {}_i$ generalized propensity score (i.e., each person treatment likelihood follow π regardless of the period)

Then, the unit-time specific effect is $\tau_{it} = Y_{it}(1) - Y_{it}(0)$

Then the Doubly Average Treatment Effect (DATE) is

$$\tau(\xi) = \sum_{T=1}^{T} \xi_t \left(\frac{1}{n} \sum_{i=1}^{n} \tau_{it} \right)$$

where

- $\frac{1}{n}\sum_{i=1}^{n} \tau_{it}$ is the unweighted effect of treatment across units (i.e., time-specific ATE).
- $\xi = (\xi_1, \dots, \xi_t)$ are user-specific weights for each time period.
- This estimand is called DATE because it's weighted (averaged) across both time and units.

A special case of DATE is when both time and unit-weights are equal

$$\tau_{eq} = \frac{1}{nT} \sum_{t=1}^{T} \sum_{i=1}^{n} \tau_{it}$$

Borrowing the idea of inverse propensity-weighted least squares estimator in the cross-sectional case that we reweight the objective function via the treatment assignment mechanism:

$$\hat{\tau} \triangleq \arg\min_{\tau} \sum_{i=1}^{n} (Y_i - \mu - W_i \tau)^2 \frac{1}{\pi_i(W_i)}$$

where

- the first term is the least squares objective
- the second term is the propensity score

In the panel data case, the IPW estimator will be

$$\hat{\tau}_{IPW} \triangleq \arg\min_{\tau} \sum_{i=1}^{n} \sum_{t=1}^{T} (Y_{it} - \alpha_i - \lambda_t - W_{it}\tau)^2 \frac{1}{\pi_i(W_i)}$$

Then, to have DATE that users can specify the structure of time weight, we use reshaped IPW estimator (ark, 2020)

$$\hat{\tau}_{RIPW}(\Pi) \triangleq \arg\min_{\tau} \sum_{i=1}^{n} \sum_{t=1}^{T} (Y_{it} - \alpha_i - \lambda_t - W_{it}\tau)^2 \frac{\Pi(W_i)}{\pi_i(W_i)}$$

where it's a function of a data-independent distribution Π that depends on the support of the treatment path $\mathbb{S} = \cup_i Supp(W_i)$

This generalization can transform to

- IPW-TWFE estimator when $\Pi \sim Unif(\mathbb{S})$
- randomized experiment when $\Pi = \pi_i$

To choose Π, we don't need to data, we just need possible assignments in your setting.

- For most practical problems (DiD, staggered, transient), we have closed form solutions
- For generic solver, we can use nonlinear programming (e..g, BFGS algorithm)

As argued in (Imai and Kim, 2021) that TWFE is not a non-parametric approach, it can be subjected to incorrect model assumption (i.e., model dependence).

- Hence, they advocate for matching methods for time-series cross-sectional data (Imai and Kim, 2021)
- Use `wfe` and `PanelMatch` to apply their paper.

This package is based on (Somaini and Wolak, 2016)

```
# dataset
library(bacondecomp)
df <- bacondecomp::castle
```

```
# devtools::install_github("paulosomaini/xtreg2way")

library(xtreg2way)
# output <- xtreg2way(y,
#                     data.frame(x1, x2),
#                     iid,
#                     tid,
#                     w,
#                     noise = "1",
#                     se = "1")

# equilvalently
output <- xtreg2way(l_homicide ~ post,
                    df,
```

```
                    iid = df$state, # group id
                    tid = df$year, # time id
                    # w, # vector of weight
                    se = "1")
output$betaHat
#>                  [,1]
#> l_homicide 0.08181162
output$aVarHat
#>              [,1]
#> [1,] 0.003396724

# to save time, you can use your structure in the
# last output for a new set of variables
# output2 <- xtreg2way(y, x1, struc=output$struc)
```

Standard errors estimation options

Set	Estimation
`se = "0"`	Assume homoskedasticity and no within group correlation or serial correlation
`se = "1"` (default)	robust to heteroskadasticity and serial correlation (Arellano, 1987)
`se = "2"`	robust to heteroskedasticity, but assumes no correlation within group or serial correlation
`se = "11"`	Aerllano SE with df correction performed by Stata xtreg (Somaini and Wolak, 2021)

Alternatively, you can also do it manually or with the `plm` package, but you have to be careful with how the SEs are estimated

```
library(multiwayvcov) # get vcov matrix
library(lmtest) # robust SEs estimation

# manual
output3 <- lm(l_homicide ~ post + factor(state) + factor(year),
              data = df)

# get variance-covariance matrix
vcov_tw <- multiwayvcov::cluster.vcov(output3,
                        cbind(df$state, df$year),
                        use_white = F,
                        df_correction = F)

# get coefficients
```

```r
coeftest(output3, vcov_tw)[2,]
#>   Estimate Std. Error    t value   Pr(>|t|)
#> 0.08181162 0.05671410 1.44252696 0.14979397
```

```r
# using the plm package
library(plm)

output4 <- plm(l_homicide ~ post,
               data = df,
               index = c("state", "year"),
               model = "within",
               effect = "twoways")

# get coefficients
coeftest(output4, vcov = vcovHC, type = "HC1")
#>
#> t test of coefficients:
#>
#>      Estimate Std. Error t value Pr(>|t|)
#> post 0.081812   0.057748  1.4167   0.1572
```

As you can see, differences stem from SE estimation, not the coefficient estimate.

29.8 Multiple periods and variation in treatment timing

This is an extension of the DiD framework to settings where you have

- more than 2 time periods
- different treatment timing

When treatment effects are heterogeneous across time or units, the standard Two-way Fixed-effects is inappropriate.

Notation is consistent with `did` package (Callaway and Sant'Anna, 2021)

- $Y_{it}(0)$ is the potential outcome for unit i
- $Y_{it}(g)$ is the potential outcome for unit i in time period t if it's treated in period g
- Y_{it} is the observed outcome for unit i in time period t

$$Y_{it} = \begin{cases} Y_{it} = Y_{it}(0) & \forall i \in \text{never-treated group} \\ Y_{it} = 1\{G_i > t\}Y_{it}(0) + 1\{G_i \leq t\}Y_{it}(G_i) & \forall i \in \text{other groups} \end{cases}$$

- G_i is the time period when i is treated
- C_i is a dummy when i belongs to the **never-treated** group
- D_{it} is a dummy for whether i is treated in period t

Assumptions:

- Staggered treatment adoption: once treated, a unit cannot be untreated (revert)
- Parallel trends assumptions (conditional on covariates):
  - Based on never-treated units: $E[Y_t(0) - Y_{t-1}(0)|G = g] = E[Y_t(0) - Y_{t-1}(0)|C = 1]$
    - Without treatment, the average potential outcomes for group g equals the average potential outcomes for the never-treated group (i.e., control group), which means that we have (1) enough data on the never-treated group (2) the control group is similar to the eventually treated group.
  - Based on not-yet treated units: $E[Y_t(0) - Y_{t-1}(0)|G = g] = E[Y_t(0) - Y_{t-1}(0)|D_s = 0, G \neq g]$
    - Not-yet treated units by time s ($s \geq t$) can be used as comparison groups to calculate the average treatment effects for the group first treated in time g
    - Additional assumption: pre-treatment trends across groups (Marcus and Sant'Anna, 2021)
- Random sampling
- Irreversibility of treatment (once treated, cannot be untreated)
- Overlap (the treatment propensity $e \in [0, 1]$)

Group-Time ATE

- This is the equivalent of the average treatment effect in the standard case (2 groups, 2 periods) under multiple time periods.

$$ATT(g, t) = E[Y_t(g) - Y_t(0)|G = g]$$

which is the average treatment effect for group g in period t

- Identification: When the parallel trends assumption based on
  - Never-treated units: $ATT(g, t) = E[Y_t - Y_{g-1}|G = g] - E[Y_t - Y_{g-1}|C = 1] \forall t \geq g$
  - Not-yet-treated units: $ATT(g, t) = E[Y_t - Y_{g-1}|G = g] - E[Y_t - Y_{g-1}|D_t = 0, G \neq g] \forall t \geq g$

- Identification: when the parallel trends assumption only holds conditional on covariates and based on
  - Never-treated units: $ATT(g,t) = E[Y_t - Y_{g-1}|X, G = g] - E[Y_t - Y_{g-1}|X, C = 1] \forall t \geq g$
  - Not-yet-treated units: $ATT(g,t) = E[Y_t - Y_{g-1}|X, G = g] - E[Y_t - Y_{g-1}|X, D_t = 0, G \neq g] \forall t \geq g$
  - This is plausible when you have suspected selection bias that can be corrected by using covariates (i.e., very much similar to matching methods to have plausible parallel trends).

Possible parameters of interest are:

1. Average treatment effect per group

$$\theta_S(g) = \frac{1}{\tau - g + 1} \sum_{t=2}^{\tau} \mathbb{1}\{\leq t\} ATT(g,t)$$

2. Average treatment effect across groups (that were treated) (similar to average treatment effect on the treated in the canonical case)

$$\theta_S^O := \sum_{g=2}^{\tau} \theta_S(g) P(G = g)$$

3. Average treatment effect dynamics (i.e., average treatment effect for groups that have been exposed to the treatment for e time periods):

$$\theta_D(e) := \sum_{g=2}^{\tau} \mathbb{1}\{g + e \leq \tau\} ATT(g, g+e) P(G = g | G + e \leq \tau)$$

4. Average treatment effect in period t for all groups that have treated by period t)

$$\theta_C(t) = \sum_{g=2}^{\tau} \mathbb{1}\{g \leq t\} ATT(g,t) P(G = g | g \leq t)$$

5. Average treatment effect by calendar time

$$\theta_C = \frac{1}{\tau - 1} \sum_{t=2}^{\tau} \theta_C(t)$$

29.9 Staggered Dif-n-dif

See Wing et al. (2024) checklist.

Recommendations by Baker et al. (2022)

- TWFE DiD regressions are suitable for single treatment periods or when treatment effects are homogeneous, provided there's a solid rationale for effect homogeneity.
- For TWFE staggered DiD, researchers should evaluate bias risks, plot treatment timings to check for variations, and use decompositions like Goodman-Bacon (2021) when possible. If decompositions aren't feasible (e.g., unbalanced panel), the percentage of never-treated units can indicate bias severity. Expected treatment effect variability should also be discussed.
- In TWFE staggered DiD event studies, avoid binning time periods without evidence of uniform effects. Use full relative-time indicators, justify reference periods, and be wary of multicollinearity causing bias.
- To address treatment timing and bias concerns, use alternative estimators like stacked regressions, Sun and Abraham (2021), Callaway and Sant'Anna (2021), or separate regressions for each event with "clean" controls.
- Justify the selection of comparison groups (not-yet treated, last treated, never treated) and ensure the parallel-trends assumption holds, especially when anticipating no effects for certain groups.

Notes:

- When subjects are treated at different point in time (variation in treatment timing across units), we have to use staggered DiD (also known as DiD event study or dynamic DiD).
- For design where a treatment is applied and units are exposed to this treatment at all time afterward, see (Athey and Imbens, 2022)

For example, basic design (Stevenson and Wolfers, 2006)

$$\begin{aligned} Y_{it} = & \sum_k \beta_k Treatment_{it}^k + \sum_i \eta_i State_i \\ & + \sum_t \lambda_t Year_t + Controls_{it} + \epsilon_{it} \end{aligned}$$

where

- $Treatment_{it}^k$ is a series of dummy variables equal to 1 if state i is treated k years ago in period t
- SE is usually clustered at the group level (occasionally time level).

- To avoid collinearity, the period right before treatment is usually chosen to drop.

The more general form of TWFE (Sun and Abraham, 2021):

First, define the relative period bin indicator as

$$D_{it}^{l} = \mathbf{1}(t - E_i = l)$$

where it's an indicator function of unit i being l periods from its first treatment at time t

1. **Static** specification

$$Y_{it} = \alpha_i + \lambda_t + \mu_g \sum_{l \geq 0} D_{it}^{l} + \epsilon_{it}$$

where

- α_i is the the unit FE
- λ_t is the time FE
- μ_g is the coefficient of interest $g = [0, T)$
- we exclude all periods before first adoption.

2. **Dynamic** specification

$$Y_{it} = \alpha_i + \lambda_t + \sum_{\substack{l=-K \\ l \neq -1}}^{L} \mu_l D_{it}^{l} + \epsilon_{it}$$

where we have to exclude some relative periods to avoid multicollinearity problem (e.g., either period right before treatment, or the treatment period).

In this setting, we try to show that the treatment and control groups are not statistically different (i.e., the coefficient estimates before treatment are not different from 0) to show pre-treatment parallel trends.

However, this two-way fixed effects design has been criticized by Sun and Abraham (2021); Callaway and Sant'Anna (2021); Goodman-Bacon (2021). When researchers include leads and lags of the treatment to see the long-term effects of the treatment, these leads and lags can be biased by effects from other periods, and pre-trends can falsely arise due to treatment effects heterogeneity.

Applying the new proposed method, finance and accounting researchers find that in many cases, the causal estimates turn out to be null (Baker et al., 2022).

Assumptions of Staggered DID

- **Rollout Exogeneity** (i.e., exogeneity of treatment adoption): if the treatment is randomly implemented over time (i.e., unrelated to variables that could also affect our dependent variables)
  - Evidence: Regress adoption on pre-treatment variables. And if you find evidence of correlation, include linear trends interacted with pre-treatment variables (Hoynes and Schanzenbach, 2009)
  - Evidence: (Deshpande and Li, 2019, p. 223)
    * Treatment is random: Regress treatment status at the unit level to all pre-treatment observables. If you have some that are predictive of treatment status, you might have to argue why it's not a worry. At best, you want this.
    * Treatment timing is random: Conditional on treatment, regress timing of the treatment on pre-treatment observables. At least, you want this.
- **No confounding events**
- **Exclusion restrictions**
  - ***No-anticipation assumption***: future treatment time do not affect current outcomes
  - ***Invariance-to-history assumption***: the time a unit under treatment does not affect the outcome (i.e., the time exposed does not matter, just whether exposed or not). This presents causal effect of early or late adoption on the outcome.
- And all the assumptions in listed in the Multiple periods and variation in treatment timing
- Auxiliary assumptions:
  - Constant treatment effects across units
  - Constant treatment effect over time
  - Random sampling
  - Effect Additivity

Remedies for staggered DiD (Baker et al., 2022):

- Each treated cohort is compared to appropriate controls (not-yet-treated, never-treated)
  - (Goodman-Bacon, 2021)
  - (Callaway and Sant'Anna, 2021) consistent for average ATT. more complicated but also more flexible than (Sun and Abraham, 2021)
    * (Sun and Abraham, 2021) (a special case of (Callaway and Sant'Anna, 2021))

  - (De Chaisemartin and d'Haultfoeuille, 2020)
  - (Borusyak et al., 2021)
- Stacked DID (biased but simple):
  - (Gormley and Matsa, 2011)
  - (Cengiz et al., 2019)
  - (Deshpande and Li, 2019)

29.9.1 Stacked DID

Notations following these slides

$$Y_{it} = \beta_{FE} D_{it} + A_i + B_t + \epsilon_{it}$$

where

- A_i is the group fixed effects
- B_t is the period fixed effects

Steps

1. Choose Event Window
2. Enumerate Sub-experiments
3. Define Inclusion Criteria
4. Stack Data
5. Specify Estimating Equation

Event Window

Let

- κ_a be the length of the pre-event window
- κ_b be the length of the post-event window

By setting a common event window for the analysis, we essentially exclude all those events that do not meet this criteria.

Sub-experiments

Let T_1 be the earliest period in the dataset

T_T be the last period in the dataset

Then, the collection of all policy adoption periods that are under our event window is

$$\Omega_A = \{A_i | T_1 + \kappa_a \leq A_i \leq T_T - \kappa_b\}$$

where these events exist

- at least κ_a periods after the earliest period
- at least κ_b periods before the last period

Let $d = 1, \dots, D$ be the index column of the sub-experiments in Ω_A

and ω_d be the event date of the d-th sub-experiment (e.g., ω_1 = adoption date of the 1st experiment)

Inclusion Criteria

1. Valid treated Units
   - Within sub-experiment d, all treated units have the same adoption date
   - This makes sure a unit can only serve as a treated unit in only 1 sub-experiment
2. Clean controls
   - Only units satisfying $A_i > \omega_d + \kappa_b$ are included as controls in sub-experiment d
   - This ensures controls are only
     - never treated units
     - units that are treated in far future
   - But a unit can be control unit in multiple sub-experiments (need to correct SE)
3. Valid Time Periods
   - All observations within sub-experiment d are from time periods within the sub-experiment's event window
   - This ensures in sub-experiment d, only observations satisfying $\omega_d - \kappa_a \leq t \leq \omega_d + \kappa_b$ are included

```r
library(did)
library(tidyverse)
library(fixest)

data(base_stagg)

# first make the stacked datasets
# get the treatment cohorts
cohorts <- base_stagg %>%
    select(year_treated) %>%
    # exclude never-treated group
    filter(year_treated != 10000) %>%
    unique() %>%
    pull()

# make formula to create the sub-datasets
getdata <- function(j, window) {
```

```
    #keep what we need
    base_stagg %>%
        # keep treated units and all units not treated within -5 to 5
        # keep treated units and all units not treated within -window to window
        filter(year_treated == j | year_treated > j + window) %>%
        # keep just year -window to window
        filter(year >= j - window & year <= j + window) %>%
        # create an indicator for the dataset
        mutate(df = j)
}

# get data stacked
stacked_data <- map_df(cohorts, ~ getdata(., window = 5)) %>%
    mutate(rel_year = if_else(df == year_treated, time_to_treatment, NA_real_)) %>%
    fastDummies::dummy_cols("rel_year", ignore_na = TRUE) %>%
    mutate(across(starts_with("rel_year_"), ~ replace_na(., 0)))

# get stacked value
stacked <-
    feols(
        y ~ `rel_year_-5` + `rel_year_-4` + `rel_year_-3` +
            `rel_year_-2` + rel_year_0 + rel_year_1 + rel_year_2 + rel_year_3 +
            rel_year_4 + rel_year_5 |
            id ^ df + year ^ df,
        data = stacked_data
    )$coefficients

stacked_se = feols(
    y ~ `rel_year_-5` + `rel_year_-4` + `rel_year_-3` +
        `rel_year_-2` + rel_year_0 + rel_year_1 + rel_year_2 + rel_year_3 +
        rel_year_4 + rel_year_5 |
        id ^ df + year ^ df,
    data = stacked_data
)$se

# add in 0 for omitted -1
stacked <- c(stacked[1:4], 0, stacked[5:10])
stacked_se <- c(stacked_se[1:4], 0, stacked_se[5:10])


cs_out <- att_gt(
    yname = "y",
    data = base_stagg,
    gname = "year_treated",
    idname = "id",
```

```
    # xformla = "~x1",
    tname = "year"
)
cs <-
    aggte(
        cs_out,
        type = "dynamic",
        min_e = -5,
        max_e = 5,
        bstrap = FALSE,
        cband = FALSE
    )



res_sa20 = feols(y ~ sunab(year_treated, year) |
                     id + year, base_stagg)
sa = tidy(res_sa20)[5:14, ] %>% pull(estimate)
sa = c(sa[1:4], 0, sa[5:10])

sa_se = tidy(res_sa20)[6:15, ] %>% pull(std.error)
sa_se = c(sa_se[1:4], 0, sa_se[5:10])

compare_df_est = data.frame(
    period = -5:5,
    cs = cs$att.egt,
    sa = sa,
    stacked = stacked
)

compare_df_se = data.frame(
    period = -5:5,
    cs = cs$se.egt,
    sa = sa_se,
    stacked = stacked_se
)

compare_df_longer <- compare_df_est %>%
    pivot_longer(!period, names_to = "estimator", values_to = "est") %>%

    full_join(compare_df_se %>%
                  pivot_longer(!period, names_to = "estimator", values_to = "se")) %>%

    mutate(upper = est +  1.96 * se,
           lower = est - 1.96 * se)
```

```r
ggplot(compare_df_longer) +
    geom_ribbon(aes(
        x = period,
        ymin = lower,
        ymax = upper,
        group = estimator
    )) +
    geom_line(aes(
        x = period,
        y = est,
        group = estimator,
        col = estimator
    ),
    linewidth = 1) +
    causalverse::ama_theme()
```

Stack Data

Estimating Equation

$$Y_{itd} = \beta_0 + \beta_1 T_{id} + \beta_2 P_{td} + \beta_3 (T_{id} \times P_{td}) + \epsilon_{itd}$$

where

- $T_{id} = 1$ if unit i is treated in sub-experiment d, 0 if control
- $P_{td} = 1$ if it's the period after the treatment in sub-experiment d

Equivalently,

$$Y_{itd} = \beta_3(T_{id} \times P_{td}) + \theta_{id} + \gamma_{td} + \epsilon_{itd}$$

β_3 averages all the time-varying effects into a single number (can't see the time-varying effects)

Stacked Event Study

Let $YSE_{td} = t - \omega_d$ be the "time since event" variable in sub-experiment d

Then, $YSE_{td} = -\kappa_a, \dots, 0, \dots, \kappa_b$ in every sub-experiment

In each sub-experiment, we can fit

$$Y_{it}^d = \sum_{j=-\kappa_a}^{\kappa_b} \beta_j^d \times 1(TSE_{td} = j) + \sum_{m=-\kappa_a}^{\kappa_b} \delta_j^d (T_{id} \times 1(TSE_{td} = j)) + \theta_i^d + \epsilon_{it}^d$$

- Different set of event study coefficients in each sub-experiment

$$Y_{itd} = \sum_{j=-\kappa_a}^{\kappa_b} \beta_j \times 1(TSE_{td} = j) + \sum_{m=-\kappa_a}^{\kappa_b} \delta_j (T_{id} \times 1(TSE_{td} = j)) + \theta_{id} + \epsilon_{itd}$$

Clustering

- Clustered at the unit x sub-experiment level (Cengiz et al., 2019)
- Clustered at the unit level (Deshpande and Li, 2019)

29.9.2 Goodman-Bacon Decomposition

Paper: (Goodman-Bacon, 2021)

For an excellent explanation slides by the author, see

Takeaways:

- A pairwise DID (τ) gets more weight if the change is close to the middle of the study window
- A pairwise DID (τ) gets more weight if it includes more observations.

Code from `bacondecomp` vignette

```
library(bacondecomp)
library(tidyverse)
data("castle")
castle <- bacondecomp::castle %>%
```

```r
    dplyr::select("l_homicide", "post", "state", "year")
head(castle)
#>   l_homicide post   state year
#> 1   2.027356    0 Alabama 2000
#> 2   2.164867    0 Alabama 2001
#> 3   1.936334    0 Alabama 2002
#> 4   1.919567    0 Alabama 2003
#> 5   1.749841    0 Alabama 2004
#> 6   2.130440    0 Alabama 2005


df_bacon <- bacon(
    l_homicide ~ post,
    data = castle,
    id_var = "state",
    time_var = "year"
)
#>                        type  weight  avg_est
#> 1 Earlier vs Later Treated 0.05976 -0.00554
#> 2 Later vs Earlier Treated 0.03190  0.07032
#> 3     Treated vs Untreated 0.90834  0.08796

# weighted average of the decomposition
sum(df_bacon$estimate * df_bacon$weight)
#> [1] 0.08181162
```

Two-way Fixed effect estimate

```r
library(broom)
fit_tw <- lm(l_homicide ~ post + factor(state) + factor(year),
             data = bacondecomp::castle)
head(tidy(fit_tw))
#> # A tibble: 6 x 5
#>   term                    estimate std.error statistic   p.value
#>   <chr>                      <dbl>     <dbl>     <dbl>     <dbl>
#> 1 (Intercept)               1.95      0.0624    31.2   2.84e-118
#> 2 post                      0.0818    0.0317     2.58  1.02e-  2
#> 3 factor(state)Alaska      -0.373     0.0797    -4.68  3.77e-  6
#> 4 factor(state)Arizona      0.0158    0.0797     0.198 8.43e-  1
#> 5 factor(state)Arkansas    -0.118     0.0810    -1.46  1.44e-  1
#> 6 factor(state)California  -0.108     0.0810    -1.34  1.82e-  1
```

Hence, naive TWFE fixed effect equals the weighted average of the Bacon decomposition (= 0.08).

```r
library(ggplot2)

ggplot(df_bacon) +
    aes(
        x = weight,
        y = estimate,
        # shape = factor(type),
        color = type
    ) +
    labs(x = "Weight", y = "Estimate", shape = "Type") +
    geom_point() +
    causalverse::ama_theme()
```

With time-varying controls that can identify variation within-treatment timing group, the"early vs. late" and "late vs. early" estimates collapse to just one estimate (i.e., both treated).

29.9.3 DID with in and out treatment condition

29.9.3.1 Panel Match

Imai and Kim (2021)

This case generalizes the staggered adoption setting, allowing units to vary in treatment over time. For N units across T time periods (with potentially unbalanced panels), let X_{it} represent treatment and Y_{it} the outcome for unit i at time t. We use the two-way linear fixed effects model:

$$Y_{it} = \alpha_i + \gamma_t + \beta X_{it} + \epsilon_{it}$$

for $i = 1, \dots, N$ and $t = 1, \dots, T$. Here, α_i and γ_t are unit and time fixed effects. They capture time-invariant unit-specific and unit-invariant time-specific unobserved confounders, respectively. We can express these as $\alpha_i = h(\mathbf{U}_i)$ and $\gamma_t = f(\mathbf{V}_t)$, with $\mathbf{U}_i$ and $\mathbf{V}_t$ being the confounders. The model doesn't assume a specific form for $h(.)$ and $f(.)$, but that they're additive and separable given binary treatment.

The least squares estimate of β leverages the covariance in outcome and treatment (Imai and Kim, 2021, p. 406). Specifically, it uses the within-unit and within-time variations. Many researchers prefer the two fixed effects (2FE) estimator because it adjusts for both types of unobserved confounders without specific functional-form assumptions, but this is wrong (Imai and Kim, 2019). We do need functional-form assumption (i.e., linearity assumption) for the 2FE to work (Imai and Kim, 2021, p. 406)

- **Two-Way Matching Estimator**:
  - It can lead to mismatches; units with the same treatment status get matched when estimating counterfactual outcomes.
  - Observations need to be matched with opposite treatment status for correct causal effects estimation.
  - Mismatches can cause attenuation bias.
  - The 2FE estimator adjusts for this bias using the factor K, which represents the net proportion of proper matches between observations with opposite treatment status.
- **Weighting in 2FE**:
  - Observation (i, t) is weighted based on how often it acts as a control unit.
  - The weighted 2FE estimator still has mismatches, but fewer than the standard 2FE estimator.
  - Adjustments are made based on observations that neither belong to the same unit nor the same time period as the matched observation.
  - This means there are challenges in adjusting for unit-specific and time-specific unobserved confounders under the two-way fixed effect framework.
- **Equivalence & Assumptions**:
  - Equivalence between the 2FE estimator and the DID estimator is dependent on the linearity assumption.

  - The multi-period DiD estimator is described as an average of two-time-period, two-group DiD estimators applied during changes from control to treatment.

- **Comparison with DiD**:
  - In simple settings (two time periods, treatment given to one group in the second period), the standard nonparametric DiD estimator equals the 2FE estimator.
  - This doesn't hold in multi-period DiD designs where units change treatment status multiple times at different intervals.
  - Contrary to popular belief, the unweighted 2FE estimator isn't generally equivalent to the multi-period DiD estimator.
  - While the multi-period DiD can be equivalent to the weighted 2FE, some control observations may have negative regression weights.

- **Conclusion**:
  - Justifying the 2FE estimator as the DID estimator isn't warranted without imposing the linearity assumption.

Application (Imai et al., 2021b)

- **Matching Methods**:
  - Enhance the validity of causal inference.
  - Reduce model dependence and provide intuitive diagnostics (Ho et al., 2007)
  - Rarely utilized in analyzing time series cross-sectional data.
  - The proposed matching estimators are more robust than the standard two-way fixed effects estimator, which can be biased if mis-specified
  - Better than synthetic controls (e.g., (Xu, 2017)) because it needs less data to achieve good performance and and adapt the the context of unit switching treatment status multiple times.

- Notes:
  - Potential carryover effects (treatment may have a long-term effect), leading to post-treatment bias.

- **Proposed Approach**:
  1. Treated observations are matched with control observations from other units in the same time period with the same treatment history up to a specified number of lags.
  2. Standard matching and weighting techniques are employed to further refine the matched set.

3. Apply a DiD estimator to adjust for time trend.

4. The goal is to have treated and matched control observations with similar covariate values.

- **Assessment**:
  - The quality of matches is evaluated through covariate balancing.

- **Estimation**:
  - Both short-term and long-term average treatment effects on the treated (ATT) are estimated.

```
library(PanelMatch)
```

Treatment Variation plot

- Visualize the variation of the treatment across space and time

- Aids in discerning whether the treatment fluctuates adequately over time and units or if the variation is primarily clustered in a subset of data.

```
DisplayTreatment(
    unit.id = "wbcode2",
    time.id = "year",
    legend.position = "none",
    xlab = "year",
    ylab = "Country Code",
    treatment = "dem",

    hide.x.tick.label = TRUE, hide.y.tick.label = TRUE,
    # dense.plot = TRUE,
    data = dem
)
```

1. Select F (i.e., the number of leads - time periods after treatment). Driven by what authors are interested in estimating:

- $F = 0$ is the contemporaneous effect (short-term effect)
- $F = n$ is the the treatment effect on the outcome two time periods after the treatment. (cumulative or long-term effect)

2. Select L (number of lags to adjust).

- Driven by the identification assumption.
- Balances bias-variance tradeoff.
- Higher L values increase credibility but reduce efficiency by limiting potential matches.

Model assumption:

- No spillover effect assumed.
- Carryover effect allowed up to L periods.
- Potential outcome for a unit depends neither on others' treatment status nor on its past treatment after L periods.

After defining causal quantity with parameters L and F.

- Focus on the average treatment effect of treatment status change.
- $\delta(F, L)$ is the average causal effect of treatment change (ATT), F periods post-treatment, considering treatment history up to L periods.
- Causal quantity considers potential future treatment reversals, meaning treatment could revert to control before outcome measurement.

Also possible to estimate the average treatment effect of treatment reversal on the reversed (ART).

Choose L, F based on specific needs.

- A large L value:
  - Increases the credibility of the limited carryover effect assumption.
  - Allows more past treatments (up to $t - L$) to influence the outcome $Y_{i,t+F}$.
  - Might reduce the number of matches and lead to less precise estimates.
- Selecting an appropriate number of lags
  - Researchers should base this choice on substantive knowledge.
  - Sensitivity of empirical results to this choice should be examined.
- The choice of F should be:
  - Substantively motivated.
  - Decides whether the interest lies in short-term or long-term causal effects.
  - A large F value can complicate causal effect interpretation, especially if many units switch treatment status during the F lead time period.

Identification Assumption

- Parallel trend assumption conditioned on treatment, outcome (excluding immediate lag), and covariate histories.
- Doesn't require strong unconfoundedness assumption.
- Cannot account for unobserved time-varying confounders.
- Essential to examine outcome time trends.
  - Check if they're parallel between treated and matched control units using pre-treatment data
- **Constructing the Matched Sets**:
  - For each treated observation, create matched control units with identical treatment history from $t - L$ to $t - 1$.
  - Matching based on treatment history helps control for carryover effects.
  - Past treatments often act as major confounders, but this method can correct for it.

  - Exact matching on time period adjusts for time-specific unobserved confounders.
  - Unlike staggered adoption methods, units can change treatment status multiple times.
  - Matched set allows treatment switching in and out of treatment
- **Refining the Matched Sets**:
  - Initially, matched sets adjust only for treatment history.
  - Parallel trend assumption requires adjustments for other confounders like past outcomes and covariates.
  - Matching methods:
    * Match each treated observation with up to J control units.
    * Distance measures like Mahalanobis distance or propensity score can be used.
    * Match based on estimated propensity score, considering pretreatment covariates.
    * Refined matched set selects most similar control units based on observed confounders.
  - Weighting methods:
    * Assign weight to each control unit in a matched set.
    * Weights prioritize more similar units.
    * Inverse propensity score weighting method can be applied.
    * Weighting is a more generalized method than matching.

The Difference-in-Differences Estimator:

- Using refined matched sets, the ATT (Average Treatment Effect on the Treated) of policy change is estimated.
- For each treated observation, estimate the counterfactual outcome using the weighted average of control units in the refined set.
- The DiD estimate of the ATT is computed for each treated observation, then averaged across all such observations.
- For noncontemporaneous treatment effects where $F > 0$:
  - The ATT doesn't specify future treatment sequence.
  - Matched control units might have units receiving treatment between time t and $t + F$.

  - Some treated units could return to control conditions between these times.

Checking Covariate Balance:

- The proposed methodology offers the advantage of checking covariate balance between treated and matched control observations.
- This check helps to see if treated and matched control observations are comparable with respect to observed confounders.
- Once matched sets are refined, covariate balance examination becomes straightforward.
- Examine the mean difference of each covariate between a treated observation and its matched controls for each pretreatment time period.
- Standardize this difference using the standard deviation of each covariate across all treated observations in the dataset.
- Aggregate this covariate balance measure across all treated observations for each covariate and pretreatment time period.
- Examine balance for lagged outcome variables over multiple pretreatment periods and time-varying covariates.
  - This helps evaluate the validity of the parallel trend assumption underlying the proposed DiD estimator.

Relations with Linear Fixed Effects Regression Estimators:

- The standard DiD estimator is equivalent to the linear two-way fixed effects regression estimator when:
  - Only two time periods exist.
  - Treatment is given to some units exclusively in the second period.
- This equivalence doesn't extend to multiperiod DiD designs, where:
  - More than two time periods are considered.
  - Units might receive treatment multiple times.
- Despite this, many researchers relate the use of the two-way fixed effects estimator to the DiD design.

Standard Error Calculation:

- Approach:
  - Condition on the weights implied by the matching process.
  - These weights denote how often an observation is utilized in matching (Imbens and Rubin, 2015)
- Context:

  - Analogous to the conditional variance seen in regression models.
  - Resulting standard errors don't factor in uncertainties around the matching procedure.
  - They can be viewed as a measure of uncertainty conditional upon the matching process (Ho et al., 2007).

Key Findings:

- Even in conditions favoring OLS, the proposed matching estimator displayed higher robustness to omitted relevant lags than the linear regression model with fixed effects.
- The robustness offered by matching came at a cost - reduced statistical power.
- This emphasizes the classic statistical tradeoff between bias (where matching has an advantage) and variance (where regression models might be more efficient).

Data Requirements

- The treatment variable is binary:
  - 0 signifies "assignment" to control.
  - 1 signifies assignment to treatment.
- Variables identifying units in the data must be: Numeric or integer.
- Variables identifying time periods should be: Consecutive numeric/integer data.
- Data format requirement: Must be provided as a standard `data.frame` object.

Basic functions:

1. Utilize treatment histories to create matching sets of treated and control units.
2. Refine these matched sets by determining weights for each control unit in the set.
   - Units with higher weights have a larger influence during estimations.

Matching on Treatment History:

- Goal is to match units transitioning from untreated to treated status with control units that have similar past treatment histories.
- Setting the Quantity of Interest (`qoi =`)
  - `att` average treatment effect on treated units
  - `atc` average treatment effect of treatment on the control units

- `art` average effect of treatment reversal for units that experience treatment reversal

- `ate` average treatment effect

```
library(PanelMatch)
# All examples follow the package's vignette
# Create the matched sets
PM.results.none <-
    PanelMatch(
        lag = 4,
        time.id = "year",
        unit.id = "wbcode2",
        treatment = "dem",
        refinement.method = "none",
        data = dem,
        match.missing = TRUE,
        size.match = 5,
        qoi = "att",
        outcome.var = "y",
        lead = 0:4,
        forbid.treatment.reversal = FALSE,
        use.diagonal.variance.matrix = TRUE
    )

# visualize the treated unit and matched controls
DisplayTreatment(
    unit.id = "wbcode2",
    time.id = "year",
    legend.position = "none",
    xlab = "year",
    ylab = "Country Code",
    treatment = "dem",
    data = dem,
    matched.set = PM.results.none$att[1],
    # highlight the particular set
    show.set.only = TRUE
)
```

Control units and the treated unit have identical treatment histories over the lag window (1988-1991)

```
DisplayTreatment(
    unit.id = "wbcode2",
    time.id = "year",
    legend.position = "none",
    xlab = "year",
    ylab = "Country Code",
    treatment = "dem",
    data = dem,
    matched.set = PM.results.none$att[2],
    # highlight the particular set
    show.set.only = TRUE
)
```

This set is more limited than the first one, but we can still see that we have exact past histories.

- **Refining Matched Sets**
  - Refinement involves assigning weights to control units.
  - Users must:
    1. Specify a method for calculating unit similarity/distance.
    2. Choose variables for similarity/distance calculations.
- **Select a Refinement Method**
  - Users determine the refinement method via the **`refinement.method`** argument.
  - Options include:
    * `mahalanobis`
    * `ps.match`
    * `CBPS.match`
    * `ps.weight`
    * `CBPS.weight`
    * `ps.msm.weight`
    * `CBPS.msm.weight`
    * `none`

  - Methods with "match" in the name and Mahalanobis will assign equal weights to similar control units.
  - "Weighting" methods give higher weights to control units more similar to treated units.

- **Variable Selection**
  - Users need to define which covariates will be used through the **`covs.formula`** argument, a one-sided formula object.
  - Variables on the right side of the formula are used for calculations.
  - "Lagged" versions of variables can be included using the format: **`I(lag(name.of.var, 0:n))`**.

- **Understanding `PanelMatch` and `matched.set` objects**
  - The **`PanelMatch` function** returns a **`PanelMatch` object**.
  - The most crucial element within the `PanelMatch` object is the **matched.set object**.
  - Within the `PanelMatch` object, the matched.set object will have names like att, art, or atc.
  - If **`qoi = ate`**, there will be two matched.set objects: att and atc.

- **Matched.set Object Details**
  - matched.set is a named list with added attributes.
  - Attributes include:
    * Lag
    * Names of treatment
    * Unit and time variables
  - Each list entry represents a matched set of treated and control units.
  - Naming follows a structure: **`[id variable].[time variable]`**.
  - Each list element is a vector of control unit ids that match the treated unit mentioned in the element name.
  - Since it's a matching method, weights are only given to the **`size.match`** most similar control units based on distance calculations.

```r
# PanelMatch without any refinement
PM.results.none <-
    PanelMatch(
        lag = 4,
        time.id = "year",
```

```
        unit.id = "wbcode2",
        treatment = "dem",
        refinement.method = "none",
        data = dem,
        match.missing = TRUE,
        size.match = 5,
        qoi = "att",
        outcome.var = "y",
        lead = 0:4,
        forbid.treatment.reversal = FALSE,
        use.diagonal.variance.matrix = TRUE
    )

# Extract the matched.set object
msets.none <- PM.results.none$att

# PanelMatch with refinement
PM.results.maha <-
    PanelMatch(
        lag = 4,
        time.id = "year",
        unit.id = "wbcode2",
        treatment = "dem",
        refinement.method = "mahalanobis", # use Mahalanobis distance
        data = dem,
        match.missing = TRUE,
        covs.formula = ~ tradewb,
        size.match = 5,
        qoi = "att" ,
        outcome.var = "y",
        lead = 0:4,
        forbid.treatment.reversal = FALSE,
        use.diagonal.variance.matrix = TRUE
    )
msets.maha <- PM.results.maha$att
```

```
# these 2 should be identical because weights are not shown
msets.none |> head()
#>   wbcode2 year matched.set.size
#> 1       4 1992               74
#> 2       4 1997                2
#> 3       6 1973               63
#> 4       6 1983               73
#> 5       7 1991               81
#> 6       7 1998                1
```

```
msets.maha |> head()
#>   wbcode2 year matched.set.size
#> 1       4 1992               74
#> 2       4 1997                2
#> 3       6 1973               63
#> 4       6 1983               73
#> 5       7 1991               81
#> 6       7 1998                1
# summary(msets.none)
# summary(msets.maha)
```

Visualizing Matched Sets with the plot method

- Users can visualize the distribution of the matched set sizes.
- A red line, by default, indicates the count of matched sets where treated units had no matching control units (i.e., empty matched sets).
- Plot adjustments can be made using `graphics::plot`.

```
plot(msets.none)
```

Comparing Methods of Refinement

- Users are encouraged to:
  - Use substantive knowledge for experimentation and evaluation.
  - Consider the following when configuring `PanelMatch`:
    1. The number of matched sets.

    2. The number of controls matched to each treated unit.
    3. Achieving covariate balance.
  - **Note**: Large numbers of small matched sets can lead to larger standard errors during the estimation stage.
  - Covariates that aren't well balanced can lead to undesirable comparisons between treated and control units.
  - Aspects to consider include:
    * Refinement method.
    * Variables for weight calculation.
    * Size of the lag window.
    * Procedures for addressing missing data (refer to **`match.missing`** and **`listwise.delete`** arguments).
    * Maximum size of matched sets (for matching methods).

- **Supportive Features:**
  - **`print`**, **`plot`**, and **`summary`** methods assist in understanding matched sets and their sizes.
  - **`get_covariate_balance`** helps evaluate covariate balance:
    * Lower values in the covariate balance calculations are preferred.

```
PM.results.none <-
    PanelMatch(
        lag = 4,
        time.id = "year",
        unit.id = "wbcode2",
        treatment = "dem",
        refinement.method = "none",
        data = dem,
        match.missing = TRUE,
        size.match = 5,
        qoi = "att",
        outcome.var = "y",
        lead = 0:4,
        forbid.treatment.reversal = FALSE,
        use.diagonal.variance.matrix = TRUE
    )
PM.results.maha <-
    PanelMatch(
        lag = 4,
        time.id = "year",
```

```
        unit.id = "wbcode2",
        treatment = "dem",
        refinement.method = "mahalanobis",
        data = dem,
        match.missing = TRUE,
        covs.formula = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
        size.match = 5,
        qoi = "att",
        outcome.var = "y",
        lead = 0:4,
        forbid.treatment.reversal = FALSE,
        use.diagonal.variance.matrix = TRUE
    )

# listwise deletion used for missing data
PM.results.listwise <-
    PanelMatch(
        lag = 4,
        time.id = "year",
        unit.id = "wbcode2",
        treatment = "dem",
        refinement.method = "mahalanobis",
        data = dem,
        match.missing = FALSE,
        listwise.delete = TRUE,
        covs.formula = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
        size.match = 5,
        qoi = "att",
        outcome.var = "y",
        lead = 0:4,
        forbid.treatment.reversal = FALSE,
        use.diagonal.variance.matrix = TRUE
    )

# propensity score based weighting method
PM.results.ps.weight <-
    PanelMatch(
        lag = 4,
        time.id = "year",
        unit.id = "wbcode2",
        treatment = "dem",
        refinement.method = "ps.weight",
        data = dem,
        match.missing = FALSE,
        listwise.delete = TRUE,
```

```
        covs.formula = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
        size.match = 5,
        qoi = "att",
        outcome.var = "y",
        lead = 0:4,
        forbid.treatment.reversal = FALSE
    )

get_covariate_balance(
    PM.results.none$att,
    data = dem,
    covariates = c("tradewb", "y"),
    plot = FALSE
)
#>         tradewb            y
#> t_4 -0.07245466  0.291871990
#> t_3 -0.20930129  0.208654876
#> t_2 -0.24425207  0.107736647
#> t_1 -0.10806125 -0.004950238

get_covariate_balance(
    PM.results.maha$att,
    data = dem,
    covariates = c("tradewb", "y"),
    plot = FALSE
)
#>         tradewb          y
#> t_4  0.04558637 0.09701606
#> t_3 -0.03312750 0.10844046
#> t_2 -0.01396793 0.08890753
#> t_1  0.10474894 0.06618865


get_covariate_balance(
    PM.results.listwise$att,
    data = dem,
    covariates = c("tradewb", "y"),
    plot = FALSE
)
#>         tradewb          y
#> t_4  0.05634922 0.05223623
#> t_3 -0.01104797 0.05217896
#> t_2  0.01411473 0.03094133
#> t_1  0.06850180 0.02092209
```

```r
get_covariate_balance(
    PM.results.ps.weight$att,
    data = dem,
    covariates = c("tradewb", "y"),
    plot = FALSE
)
#>         tradewb          y
#> t_4 0.014362590 0.04035905
#> t_3 0.005529734 0.04188731
#> t_2 0.009410044 0.04195008
#> t_1 0.027907540 0.03975173
```

get__covariate__balance Function Options:

- Allows for the generation of plots displaying covariate balance using **`plot = TRUE`**.

- Plots can be customized using arguments typically used with the base R **`plot`** method.

- Option to set **`use.equal.weights = TRUE`** for:

  - Obtaining the balance of unrefined sets.

  - Facilitating understanding of the refinement's impact.

```r
# Use equal weights
get_covariate_balance(
    PM.results.ps.weight$att,
    data = dem,
    use.equal.weights = TRUE,
    covariates = c("tradewb", "y"),
    plot = TRUE,
    # visualize by setting plot to TRUE
    ylim = c(-1, 1)
)
```

```
# Compare covariate balance to refined sets
# See large improvement in balance
get_covariate_balance(
    PM.results.ps.weight$att,
    data = dem,
    covariates = c("tradewb", "y"),
    plot = TRUE,
    # visualize by setting plot to TRUE
    ylim = c(-1, 1)
)
```

```
balance_scatter(
    matched_set_list = list(PM.results.maha$att,
                            PM.results.ps.weight$att),
    data = dem,
    covariates = c("y", "tradewb")
)
```

`PanelEstimate`

- **Standard Error Calculation Methods**
  - There are different methods available:
    - **Bootstrap** (default method with 1000 iterations).
    - **Conditional**: Assumes independence across units, but not time.
    - **Unconditional**: Doesn't make assumptions of independence across units or time.
  - For `qoi` values set to `att`, `art`, or `atc` (Imai et al., 2021b):
    - You can use analytical methods for calculating standard errors, which include both "conditional" and "unconditional" methods.

```
PE.results <- PanelEstimate(
    sets              = PM.results.ps.weight,
    data              = dem,
    se.method         = "bootstrap",
    number.iterations = 1000,
    confidence.level  = .95
)

# point estimates
PE.results[["estimates"]]
#>       t+0       t+1       t+2       t+3       t+4
#> 0.2609565 0.9630847 1.2851017 1.7370930 1.4871846

# standard errors
PE.results[["standard.error"]]
#>       t+0       t+1       t+2       t+3       t+4
#> 0.6399349 1.0304938 1.3825265 1.7625951 2.1672629


# use conditional method
PE.results <- PanelEstimate(
    sets             = PM.results.ps.weight,
    data             = dem,
    se.method        = "conditional",
    confidence.level = .95
)

# point estimates
PE.results[["estimates"]]
#>       t+0       t+1       t+2       t+3       t+4
#> 0.2609565 0.9630847 1.2851017 1.7370930 1.4871846
```

```r
# standard errors
PE.results[["standard.error"]]
#>       t+0       t+1       t+2       t+3       t+4
#> 0.4844805 0.8170604 1.1171942 1.4116879 1.7172143

summary(PE.results)
#> Weighted Difference-in-Differences with Propensity Score
#> Matches created with 4 lags
#>
#> Standard errors computed with conditional  method
#>
#> Estimate of Average Treatment Effect on the Treated (ATT) by Period:
#> $summary
#>      estimate std.error       2.5%    97.5%
#> t+0 0.2609565 0.4844805 -0.6886078 1.210521
#> t+1 0.9630847 0.8170604 -0.6383243 2.564494
#> t+2 1.2851017 1.1171942 -0.9045586 3.474762
#> t+3 1.7370930 1.4116879 -1.0297644 4.503950
#> t+4 1.4871846 1.7172143 -1.8784937 4.852863
#>
#> $lag
#> [1] 4
#>
#> $qoi
#> [1] "att"

plot(PE.results)
```

Moderating Variables

```
# moderating variable
dem$moderator <- 0
dem$moderator <- ifelse(dem$wbcode2 > 100, 1, 2)

PM.results <-
    PanelMatch(
        lag                          = 4,
        time.id                      = "year",
        unit.id                      = "wbcode2",
        treatment                    = "dem",
        refinement.method            = "mahalanobis",
        data                         = dem,
        match.missing                = TRUE,
        covs.formula                 = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
        size.match                   = 5,
        qoi                          = "att",
        outcome.var                  = "y",
        lead                         = 0:4,
        forbid.treatment.reversal    = FALSE,
        use.diagonal.variance.matrix = TRUE
    )
PE.results <-
    PanelEstimate(sets      = PM.results,
                  data      = dem,
                  moderator = "moderator")
```

```
# Each element in the list corresponds to a level in the moderator
plot(PE.results[[1]])
```

```
plot(PE.results[[2]])
```

To write up for journal submission, you can follow the following report:

In this study, closely aligned with the research by (Acemoglu et al., 2019), two

key effects of democracy on economic growth are estimated: the impact of democratization and that of authoritarian reversal. The treatment variable, X_{it}, is defined to be one if country i is democratic in year t, and zero otherwise.

The Average Treatment Effect for the Treated (ATT) under democratization is formulated as follows:

$$\begin{aligned}\delta(F,L) = \mathbb{E}\big\{&Y_{i,t+F}(X_{it}=1, X_{i,t-1}=0, \{X_{i,t-l}\}_{l=2}^{L}) \\ &-Y_{i,t+F}(X_{it}=0, X_{i,t-1}=0, \{X_{i,t-l}\}_{l=2}^{L}) | X_{it}=1, X_{i,t-1}=0\big\}\end{aligned}$$

In this framework, the treated observations are countries that transition from an authoritarian regime $X_{it-1} = 0$ to a democratic one $X_{it} = 1$. The variable F represents the number of leads, denoting the time periods following the treatment, and L signifies the number of lags, indicating the time periods preceding the treatment.

The ATT under authoritarian reversal is given by:

$$\begin{aligned}&\mathbb{E}\big[Y_{i,t+F}(X_{it}=0, X_{i,t-1}=1, \{X_{i,t-l}\}_{l=2}^{L}) \\ &-Y_{i,t+F}(X_{it}=1, X_{it-1}=1, \{X_{i,t-l}\}_{l=2}^{L}) | X_{it}=0, X_{i,t-1}=1\big]\end{aligned}$$

The ATT is calculated conditioning on 4 years of lags ($L = 4$) and up to 4 years following the policy change $F = 1, 2, 3, 4$. Matched sets for each treated observation are constructed based on its treatment history, with the number of matched control units generally decreasing when considering a 4-year treatment history as compared to a 1-year history.

To enhance the quality of matched sets, methods such as Mahalanobis distance matching, propensity score matching, and propensity score weighting are utilized. These approaches enable us to evaluate the effectiveness of each refinement method. In the process of matching, we employ both up-to-five and up-to-ten matching to investigate how sensitive our empirical results are to the maximum number of allowed matches. For more information on the refinement process, please see the Web Appendix

> The Mahalanobis distance is expressed through a specific formula. We aim to pair each treated unit with a maximum of J control units, permitting replacement, denoted as $|\mathcal{M}_{it} \leq J|$. The average Mahalanobis distance between a treated and each control unit over time is computed as:
>
> $$S_{it}(i') = \frac{1}{L}\sum_{l=1}^{L}\sqrt{(\mathbf{V}_{i,t-l} - \mathbf{V}_{i',t-l})^T \,{}^{-1}_{i,t-l}(\mathbf{V}_{i,t-l} - \mathbf{V}_{i',t-l})}$$
>
> For a matched control unit $i' \in \mathcal{M}_{it}$, $\mathbf{V}_{it'}$ represents the time-varying covariates to adjust for, and ${}_{it'}$ is the sample covariance

matrix for $\mathbf{V}_{it'}$. Essentially, we calculate a standardized distance using time-varying covariates and average this across different time intervals.

In the context of propensity score matching, we employ a logistic regression model with balanced covariates to derive the propensity score. Defined as the conditional likelihood of treatment given pre-treatment covariates (Rosenbaum and Rubin, 1983), the propensity score is estimated by first creating a data subset comprised of all treated and their matched control units from the same year. This logistic regression model is then fitted as follows:

$$\begin{aligned} & e_{it}(\{\mathbf{U}_{i,t-l}\}_{l=1}^{L}) \\ & = Pr(X_{it} = 1 | \mathbf{U}_{i,t-1}, \dots, \mathbf{U}_{i,t-L}) \\ & = \frac{1}{1 = \exp(-\sum_{l=1}^{L} \beta_l^T \mathbf{U}_{i,t-l})} \end{aligned}$$

where $\mathbf{U}_{it'} = (X_{it'}, \mathbf{V}_{it'}^T)^T$. Given this model, the estimated propensity score for all treated and matched control units is then computed. This enables the adjustment for lagged covariates via matching on the calculated propensity score, resulting in the following distance measure:

$$S_{it}(i') = |\text{logit}\{\hat{e}_{it}(\{\mathbf{U}_{i,t-l}\}_{l=1}^{L})\} - \text{logit}\{\hat{e}_{i't}(\{\mathbf{U}_{i',t-l}\}_{l=1}^{L})\}|$$

Here, $\hat{e}_{i't}(\{\mathbf{U}_{i,t-l}\}_{l=1}^{L})$ represents the estimated propensity score for each matched control unit $i' \in \mathcal{M}_{it}$.

Once the distance measure $S_{it}(i')$ has been determined for all control units in the original matched set, we fine-tune this set by selecting up to J closest control units, which meet a researcher-defined caliper constraint C. All other control units receive zero weight. This results in a refined matched set for each treated unit (i, t):

$$\mathcal{M}_{it}^{*} = \{i' : i' \in \mathcal{M}_{it}, S_{it}(i') < C, S_{it} \leq S_{it}^{(J)}\}$$

$S_{it}^{(J)}$ is the Jth smallest distance among the control units in the original set $\mathcal{M}_{it}$.

For further refinement using weighting, a weight is assigned to each control unit i' in a matched set corresponding to a treated unit (i, t), with greater weight accorded to more similar units. We utilize inverse propensity score weighting, based on the propensity score model mentioned earlier:

$$w_{it}^{i'} \propto \frac{\hat{e}_{i't}(\{\mathbf{U}_{i,t-l}\}_{l=1}^{L})}{1-\hat{e}_{i't}(\{\mathbf{U}_{i,t-l}\}_{l=1}^{L})}$$

In this model, $\sum_{i' \in \mathcal{M}_{it}} w_{it}^{i'} = 1$ and $w_{it}^{i'} = 0$ for $i' \notin \mathcal{M}_{it}$. The model is fitted to the complete sample of treated and matched control units.

Checking Covariate Balance A distinct advantage of the proposed methodology over regression methods is the ability it offers researchers to inspect the covariate balance between treated and matched control observations. This facilitates the evaluation of whether treated and matched control observations are comparable regarding observed confounders. To investigate the mean difference of each covariate (e.g., $V_{it'j}$, representing the j-th variable in $\mathbf{V}_{it'}$) between the treated observation and its matched control observation at each pre-treatment time period (i.e., $t' < t$), we further standardize this difference. For any given pretreatment time period, we adjust by the standard deviation of each covariate across all treated observations in the dataset. Thus, the mean difference is quantified in terms of standard deviation units. Formally, for each treated observation (i,t) where $D_{it} = 1$, we define the covariate balance for variable j at the pretreatment time period $t-l$ as:

$$B_{it}(j,l) = \frac{V_{i,t-l,j} - \sum_{i' \in \mathcal{M}_{it}} w_{it}^{i'} V_{i',t-l,j}}{\sqrt{\frac{1}{N_1-1}\sum_{i'=1}^{N}\sum_{t'=L+1}^{T-F} D_{i't'}(V_{i',t'-l,j} - \bar{V}_{t'-l,j})^2}} \quad (29.1)$$

where $N_1 = \sum_{i'=1}^{N}\sum_{t'=L+1}^{T-F} D_{i't'}$ denotes the total number of treated observations and $\bar{V}_{t-l,j} = \sum_{i=1}^{N} D_{i,t-l,j}/N$. We then aggregate this covariate balance measure across all treated observations for each covariate and pre-treatment time period:

$$\bar{B}(j,l) = \frac{1}{N_1}\sum_{i=1}^{N}\sum_{t=L+1}^{T-F} D_{it}B_{it}(j,l) \quad (29.2)$$

Lastly, we evaluate the balance of lagged outcome variables over several pre-treatment periods and that of time-varying covariates. This examination aids in assessing the validity of the parallel trend assumption integral to the DiD estimator justification.

In Figure **??**, we demonstrate the enhancement of covariate balance thank to the refinement of matched sets. Each scatter plot contrasts the absolute standardized mean difference, as detailed in Equation (29.2), before (horizontal axis) and after (vertical axis) this refinement. Points below the 45-degree line indicate an improved standardized mean balance for certain time-varying covariates post-refinement. The majority of variables benefit from this refinement process.

Notably, the propensity score weighting (bottom panel) shows the most significant improvement, whereas Mahalanobis matching (top panel) yields a more modest improvement.

```
library(PanelMatch)
library(causalverse)

runPanelMatch <- function(method, lag, size.match=NULL, qoi="att") {

    # Default parameters for PanelMatch
    common.args <- list(
        lag = lag,
        time.id = "year",
        unit.id = "wbcode2",
        treatment = "dem",
        data = dem,
        covs.formula = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
        qoi = qoi,
        outcome.var = "y",
        lead = 0:4,
        forbid.treatment.reversal = FALSE,
        size.match = size.match  # setting size.match here for all methods
    )

    if(method == "mahalanobis") {
        common.args$refinement.method <- "mahalanobis"
        common.args$match.missing <- TRUE
        common.args$use.diagonal.variance.matrix <- TRUE
    } else if(method == "ps.match") {
        common.args$refinement.method <- "ps.match"
        common.args$match.missing <- FALSE
        common.args$listwise.delete <- TRUE
    } else if(method == "ps.weight") {
        common.args$refinement.method <- "ps.weight"
        common.args$match.missing <- FALSE
        common.args$listwise.delete <- TRUE
    }

    return(do.call(PanelMatch, common.args))
}

methods <- c("mahalanobis", "ps.match", "ps.weight")
lags <- c(1, 4)
sizes <- c(5, 10)
```

You can either do it sequentailly

```r
res_pm <- list()

for(method in methods) {
    for(lag in lags) {
        for(size in sizes) {
            name <- paste0(method, ".", lag, "lag.", size, "m")
            res_pm[[name]] <- runPanelMatch(method, lag, size)
        }
    }
}

# Now, you can access res_pm using res_pm[["mahalanobis.1lag.5m"]] etc.

# for treatment reversal
res_pm_rev <- list()

for(method in methods) {
    for(lag in lags) {
        for(size in sizes) {
            name <- paste0(method, ".", lag, "lag.", size, "m")
            res_pm_rev[[name]] <- runPanelMatch(method, lag, size, qoi = "art")
        }
    }
}
```

or in parallel

```r
library(foreach)
library(doParallel)
registerDoParallel(cores = 4)
# Initialize an empty list to store results
res_pm <- list()

# Replace nested for-loops with foreach
results <-
  foreach(
    method = methods,
    .combine = 'c',
    .multicombine = TRUE,
    .packages = c("PanelMatch", "causalverse")
  ) %dopar% {
    tmp <- list()
    for (lag in lags) {
      for (size in sizes) {
        name <- paste0(method, ".", lag, "lag.", size, "m")
        tmp[[name]] <- runPanelMatch(method, lag, size)
```

```
      }
    }
    tmp
  }

# Collate results
for (name in names(results)) {
  res_pm[[name]] <- results[[name]]
}

# Treatment reversal
# Initialize an empty list to store results
res_pm_rev <- list()

# Replace nested for-loops with foreach
results_rev <-
  foreach(
    method = methods,
    .combine = 'c',
    .multicombine = TRUE,
    .packages = c("PanelMatch", "causalverse")
  ) %dopar% {
    tmp <- list()
    for (lag in lags) {
      for (size in sizes) {
        name <- paste0(method, ".", lag, "lag.", size, "m")
        tmp[[name]] <-
          runPanelMatch(method, lag, size, qoi = "art")
      }
    }
    tmp
  }

# Collate results
for (name in names(results_rev)) {
  res_pm_rev[[name]] <- results_rev[[name]]
}


stopImplicitCluster()
```

```
library(gridExtra)

# Updated plotting function
create_balance_plot <- function(method, lag, sizes, res_pm, dem) {
```

```
    matched_set_lists <- lapply(sizes, function(size) {
        res_pm[[paste0(method, ".", lag, "lag.", size, "m")]]$att
    })

    return(
        balance_scatter_custom(
            matched_set_list = matched_set_lists,
            legend.title = "Possible Matches",
            set.names = as.character(sizes),
            legend.position = c(0.2, 0.8),

            # for compiled plot, you don't need x,y, or main labs
            x.axis.label = "",
            y.axis.label = "",
            main = "",
            data = dem,
            dot.size = 5,
            # show.legend = F,
            them_use = causalverse::ama_theme(base_size = 32),
            covariates = c("y", "tradewb")
        )
    )
}

plots <- list()

for (method in methods) {
    for (lag in lags) {
        plots[[paste0(method, ".", lag, "lag")]] <-
            create_balance_plot(method, lag, sizes, res_pm, dem)
    }
}

# # Arranging plots in a 3x2 grid
# grid.arrange(plots[["mahalanobis.1lag"]],
#              plots[["mahalanobis.4lag"]],
#              plots[["ps.match.1lag"]],
#              plots[["ps.match.4lag"]],
#              plots[["ps.weight.1lag"]],
#              plots[["ps.weight.4lag"]],
#              ncol=2, nrow=3)


# Standardized Mean Difference of Covariates
library(gridExtra)
```

```r
library(grid)

# Create column and row labels using textGrob
col_labels <- c("1-year Lag", "4-year Lag")
row_labels <- c("Maha Matching", "PS Matching", "PS Weigthing")

major.axes.fontsize = 40
minor.axes.fontsize = 30

png(
    file.path(getwd(), "images", "did_balance_scatter.png"),
    width = 1200,
    height = 1000
)

# Create a list-of-lists, where each inner list represents a row
grid_list <- list(
    list(
        nullGrob(),
        textGrob(col_labels[1], gp = gpar(fontsize = minor.axes.fontsize)),
        textGrob(col_labels[2], gp = gpar(fontsize = minor.axes.fontsize))
    ),

    list(textGrob(
        row_labels[1],
        gp = gpar(fontsize = minor.axes.fontsize),
        rot = 90
    ), plots[["mahalanobis.1lag"]], plots[["mahalanobis.4lag"]]),

    list(textGrob(
        row_labels[2],
        gp = gpar(fontsize = minor.axes.fontsize),
        rot = 90
    ), plots[["ps.match.1lag"]], plots[["ps.match.4lag"]]),

    list(textGrob(
        row_labels[3],
        gp = gpar(fontsize = minor.axes.fontsize),
        rot = 90
    ), plots[["ps.weight.1lag"]], plots[["ps.weight.4lag"]])
)

# "Flatten" the list-of-lists into a single list of grobs
grobs <- do.call(c, grid_list)
```

```r
grid.arrange(
    grobs = grobs,
    ncol = 3,
    nrow = 4,
    widths = c(0.15, 0.42, 0.42),
    heights = c(0.15, 0.28, 0.28, 0.28)
)

grid.text(
    "Before Refinement",
    x = 0.5,
    y = 0.03,
    gp = gpar(fontsize = major.axes.fontsize)
)
grid.text(
    "After Refinement",
    x = 0.03,
    y = 0.5,
    rot = 90,
    gp = gpar(fontsize = major.axes.fontsize)
)
dev.off()
#> pdf
#>   2
```

Note: Scatter plots display the standardized mean difference of each covariate j and lag year l as defined in Equation (29.2) before (x-axis) and after (y-axis) matched set refinement. Each plot includes varying numbers of possible matches for each matching method. Rows represent different matching/weighting methods, while columns indicate adjustments for various lag lengths.

```r
# Step 1: Define configurations
configurations <- list(
    list(refinement.method = "none", qoi = "att"),
    list(refinement.method = "none", qoi = "art"),
    list(refinement.method = "mahalanobis", qoi = "att"),
    list(refinement.method = "mahalanobis", qoi = "art"),
    list(refinement.method = "ps.match", qoi = "att"),
    list(refinement.method = "ps.match", qoi = "art"),
    list(refinement.method = "ps.weight", qoi = "att"),
    list(refinement.method = "ps.weight", qoi = "art")
)

# Step 2: Use lapply or loop to generate results
results <- lapply(configurations, function(config) {
    PanelMatch(
```

```
        lag                        = 4,
        time.id                    = "year",
        unit.id                    = "wbcode2",
        treatment                  = "dem",
        data                       = dem,
        match.missing              = FALSE,
        listwise.delete            = TRUE,
        size.match                 = 5,
        outcome.var                = "y",
        lead                       = 0:4,
        forbid.treatment.reversal = FALSE,
        refinement.method          = config$refinement.method,
        covs.formula               = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
        qoi                        = config$qoi
    )
})

# Step 3: Get covariate balance and plot
plots <- mapply(function(result, config) {
    df <- get_covariate_balance(
        if (config$qoi == "att")
            result$att
        else
            result$art,
        data = dem,
        covariates = c("tradewb", "y"),
        plot = F
    )
    causalverse::plot_covariate_balance_pretrend(df, main = "", show_legend = F)
}, results, configurations, SIMPLIFY = FALSE)

# Set names for plots
names(plots) <- sapply(configurations, function(config) {
    paste(config$qoi, config$refinement.method, sep = ".")
})
```

To export

```
library(gridExtra)
library(grid)

# Column and row labels
col_labels <-
    c("None",
      "Mahalanobis",
      "Propensity Score Matching",
```

```
      "Propensity Score Weighting")
row_labels <- c("ATT", "ART")

# Specify your desired fontsize for labels
minor.axes.fontsize <- 16
major.axes.fontsize <- 20

png(file.path(getwd(), "images", "p_covariate_balance.png"), width=1200, height=1000)

# Create a list-of-lists, where each inner list represents a row
grid_list <- list(
    list(
        nullGrob(),
        textGrob(col_labels[1], gp = gpar(fontsize = minor.axes.fontsize)),
        textGrob(col_labels[2], gp = gpar(fontsize = minor.axes.fontsize)),
        textGrob(col_labels[3], gp = gpar(fontsize = minor.axes.fontsize)),
        textGrob(col_labels[4], gp = gpar(fontsize = minor.axes.fontsize))
    ),

    list(
        textGrob(
            row_labels[1],
            gp = gpar(fontsize = minor.axes.fontsize),
            rot = 90
        ),
        plots$att.none,
        plots$att.mahalanobis,
        plots$att.ps.match,
        plots$att.ps.weight
    ),

    list(
        textGrob(
            row_labels[2],
            gp = gpar(fontsize = minor.axes.fontsize),
            rot = 90
        ),
        plots$art.none,
        plots$art.mahalanobis,
        plots$art.ps.match,
        plots$art.ps.weight
    )
)

# "Flatten" the list-of-lists into a single list of grobs
```

```r
grobs <- do.call(c, grid_list)

# Arrange your plots with text labels
grid.arrange(
    grobs   = grobs,
    ncol    = 5,
    nrow    = 3,
    widths  = c(0.1, 0.225, 0.225, 0.225, 0.225),
    heights = c(0.1, 0.45, 0.45)
)

# Add main x and y axis titles
grid.text(
    "Refinement Methods",
    x  = 0.5,
    y  = 0.01,
    gp = gpar(fontsize = major.axes.fontsize)
)
grid.text(
    "Quantities of Interest",
    x   = 0.02,
    y   = 0.5,
    rot = 90,
    gp  = gpar(fontsize = major.axes.fontsize)
)

dev.off()
```

```r
library(knitr)
include_graphics(file.path(getwd(), "images", "p_covariate_balance.png"))
```

Note: Each graph displays the standardized mean difference, as outlined in Equation (29.2), plotted on the vertical axis across a pre-treatment duration of four years represented on the horizontal axis. The leftmost column illustrates the balance prior to refinement, while the subsequent three columns depict the covariate balance post the application of distinct refinement techniques. Each individual line signifies the balance of a specific variable during the pre-treatment phase.The red line is tradewb and blue line is the lagged outcome variable.

In Figure **??**, we observe a marked improvement in covariate balance due to the implemented matching procedures during the pre-treatment period. Our analysis prioritizes methods that adjust for time-varying covariates over a span of four years preceding the treatment initiation. The two rows delineate the standardized mean balance for both treatment modalities, with individual lines representing the balance for each covariate.

Across all scenarios, the refinement attributed to matched sets significantly enhances balance. Notably, using propensity score weighting considerably mitigates imbalances in confounders. While some degree of imbalance remains evident in the Mahalanobis distance and propensity score matching techniques, the standardized mean difference for the lagged outcome remains stable throughout the pre-treatment phase. This consistency lends credence to the validity of the proposed DiD estimator.

Estimation Results

We now detail the estimated ATTs derived from the matching techniques. Figure below offers visual representations of the impacts of treatment initiation (upper panel) and treatment reversal (lower panel) on the outcome variable for a duration of 5 years post-transition, specifically, (F = 0, 1, ..., 4). Across the five methods (columns), it becomes evident that the point estimates of effects associated with treatment initiation consistently approximate zero over the 5-year window. In contrast, the estimated outcomes of treatment reversal are notably negative and maintain statistical significance through all refinement techniques during the initial year of transition and the 1 to 4 years that follow, provided treatment reversal is permissible. These effects are notably pronounced, pointing to an estimated reduction of roughly X% in the outcome variable.

Collectively, these findings indicate that the transition into the treated state from its absence doesn't invariably lead to a heightened outcome. Instead, the transition from the treated state back to its absence exerts a considerable negative effect on the outcome variable in both the short and intermediate terms. Hence, the positive effect of the treatment (if we were to use traditional DiD) is actually driven by the negative effect of treatment reversal.

```
# sequential
# Step 1: Apply PanelEstimate function

# Initialize an empty list to store results
res_est <- vector("list", length(res_pm))

# Iterate over each element in res_pm
for (i in 1:length(res_pm)) {
  res_est[[i]] <- PanelEstimate(
    res_pm[[i]],
    data = dem,
    se.method = "bootstrap",
    number.iterations = 1000,
    confidence.level = .95
  )
  # Transfer the name of the current element to the res_est list
  names(res_est)[i] <- names(res_pm)[i]
}

# Step 2: Apply plot_PanelEstimate function

# Initialize an empty list to store plot results
res_est_plot <- vector("list", length(res_est))

# Iterate over each element in res_est
for (i in 1:length(res_est)) {
    res_est_plot[[i]] <-
```

```
        plot_PanelEstimate(res_est[[i]],
                           main = "",
                           theme_use = causalverse::ama_theme(base_size = 14))
    # Transfer the name of the current element to the res_est_plot list
    names(res_est_plot)[i] <- names(res_est)[i]
}

# check results
# res_est_plot$mahalanobis.1lag.5m


# Step 1: Apply PanelEstimate function for res_pm_rev

# Initialize an empty list to store results
res_est_rev <- vector("list", length(res_pm_rev))

# Iterate over each element in res_pm_rev
for (i in 1:length(res_pm_rev)) {
  res_est_rev[[i]] <- PanelEstimate(
    res_pm_rev[[i]],
    data = dem,
    se.method = "bootstrap",
    number.iterations = 1000,
    confidence.level = .95
  )
  # Transfer the name of the current element to the res_est_rev list
  names(res_est_rev)[i] <- names(res_pm_rev)[i]
}

# Step 2: Apply plot_PanelEstimate function for res_est_rev

# Initialize an empty list to store plot results
res_est_plot_rev <- vector("list", length(res_est_rev))

# Iterate over each element in res_est_rev
for (i in 1:length(res_est_rev)) {
    res_est_plot_rev[[i]] <-
        plot_PanelEstimate(res_est_rev[[i]],
                           main = "",
                           theme_use = causalverse::ama_theme(base_size = 14))
  # Transfer the name of the current element to the res_est_plot_rev list
  names(res_est_plot_rev)[i] <- names(res_est_rev)[i]
}
```

```r
# parallel
library(doParallel)
library(foreach)

# Detect the number of cores to use for parallel processing
num_cores <- 4

# Register the parallel backend
cl <- makeCluster(num_cores)
registerDoParallel(cl)

# Step 1: Apply PanelEstimate function in parallel
res_est <-
    foreach(i = 1:length(res_pm), .packages = "PanelMatch") %dopar% {
        PanelEstimate(
            res_pm[[i]],
            data = dem,
            se.method = "bootstrap",
            number.iterations = 1000,
            confidence.level = .95
        )
    }

# Transfer names from res_pm to res_est
names(res_est) <- names(res_pm)

# Step 2: Apply plot_PanelEstimate function in parallel
res_est_plot <-
    foreach(
        i = 1:length(res_est),
        .packages = c("PanelMatch", "causalverse", "ggplot2")
    ) %dopar% {
        plot_PanelEstimate(res_est[[i]],
                           main = "",
                           theme_use = causalverse::ama_theme(base_size = 10))
    }

# Transfer names from res_est to res_est_plot
names(res_est_plot) <- names(res_est)



# Step 1: Apply PanelEstimate function for res_pm_rev in parallel
res_est_rev <-
    foreach(i = 1:length(res_pm_rev), .packages = "PanelMatch") %dopar% {
```

```
        PanelEstimate(
            res_pm_rev[[i]],
            data = dem,
            se.method = "bootstrap",
            number.iterations = 1000,
            confidence.level = .95
        )
    }

# Transfer names from res_pm_rev to res_est_rev
names(res_est_rev) <- names(res_pm_rev)

# Step 2: Apply plot_PanelEstimate function for res_est_rev in parallel
res_est_plot_rev <-
    foreach(
        i = 1:length(res_est_rev),
        .packages = c("PanelMatch", "causalverse", "ggplot2")
    ) %dopar% {
        plot_PanelEstimate(res_est_rev[[i]],
                           main = "",
                           theme_use = causalverse::ama_theme(base_size = 10))
    }

# Transfer names from res_est_rev to res_est_plot_rev
names(res_est_plot_rev) <- names(res_est_rev)

# Stop the cluster
stopCluster(cl)
```

To export

```
library(gridExtra)
library(grid)

# Column and row labels
col_labels <- c("Mahalanobis 5m",
                "Mahalanobis 10m",
                "PS Matching 5m",
                "PS Matching 10m",
                "PS Weighting 5m")

row_labels <- c("ATT", "ART")

# Specify your desired fontsize for labels
minor.axes.fontsize <- 16
major.axes.fontsize <- 20
```

```r
png(file.path(getwd(), "images", "p_did_est_in_n_out.png"), width=1200, height=1000)

# Create a list-of-lists, where each inner list represents a row
grid_list <- list(
  list(
    nullGrob(),
    textGrob(col_labels[1], gp = gpar(fontsize = minor.axes.fontsize)),
    textGrob(col_labels[2], gp = gpar(fontsize = minor.axes.fontsize)),
    textGrob(col_labels[3], gp = gpar(fontsize = minor.axes.fontsize)),
    textGrob(col_labels[4], gp = gpar(fontsize = minor.axes.fontsize)),
    textGrob(col_labels[5], gp = gpar(fontsize = minor.axes.fontsize))
  ),

  list(
    textGrob(row_labels[1], gp = gpar(fontsize = minor.axes.fontsize), rot = 90),
    res_est_plot$mahalanobis.1lag.5m,
    res_est_plot$mahalanobis.1lag.10m,
    res_est_plot$ps.match.1lag.5m,
    res_est_plot$ps.match.1lag.10m,
    res_est_plot$ps.weight.1lag.5m
  ),

  list(
    textGrob(row_labels[2], gp = gpar(fontsize = minor.axes.fontsize), rot = 90),
    res_est_plot_rev$mahalanobis.1lag.5m,
    res_est_plot_rev$mahalanobis.1lag.10m,
    res_est_plot_rev$ps.match.1lag.5m,
    res_est_plot_rev$ps.match.1lag.10m,
    res_est_plot_rev$ps.weight.1lag.5m
  )
)

# "Flatten" the list-of-lists into a single list of grobs
grobs <- do.call(c, grid_list)

# Arrange your plots with text labels
grid.arrange(
  grobs   = grobs,
  ncol    = 6,
  nrow    = 3,
  widths  = c(0.1, 0.18, 0.18, 0.18, 0.18, 0.18),
  heights = c(0.1, 0.45, 0.45)
)

# Add main x and y axis titles
```

```
grid.text(
  "Methods",
  x  = 0.5,
  y  = 0.02,
  gp = gpar(fontsize = major.axes.fontsize)
)
grid.text(
  "",
  x   = 0.02,
  y   = 0.5,
  rot = 90,
  gp  = gpar(fontsize = major.axes.fontsize)
)

dev.off()
```

```
library(knitr)
include_graphics(file.path(getwd(), "images", "p_did_est_in_n_out.png"))
```

29.9.3.2 Counterfactual Estimators

- Also known as **imputation approach** (Liu et al., 2022)
- This class of estimator consider observation treatment as missing data. Models are built using data from the control units to impute conterfactuals for the treated observations.
- It's called counterfactual estimators because they predict outcomes as if the treated observations had not received the treatment.
- Advantages:
  - Avoids negative weights and biases by not using treated observations for modeling and applying uniform weights.
  - Supports various models, including those that may relax strict exogeneity assumptions.
- Methods including
  - Fixed-effects conterfactual estimator (FEct) (DiD is a special case):
    * Based on the Two-way Fixed-effects, where assumes linear additive functional form of unobservables based on unit and time FEs. But FEct fixes the improper weighting of TWFE by comparing within each matched pair (where each pair is the treated observation and its predicted counterfactual that is the weighted sum of all untreated observations).
  - Interactive Fixed Effects conterfactual estimator (IFEct) (Gobillon and Magnac, 2016, Xu (2017)):
    * When we suspect unobserved time-varying confounder, FEct fails. Instead, IFEct uses the factor-augmented models to relax the strict exogeneity assumption where the effects of

unobservables can be decomposed to unit FE + time FE + unit x time FE.
    * Generalized Synthetic Controls are a subset of IFEct when treatments don't revert.
  - Matrix completion (MC) (Athey et al., 2021):
    * Generalization of factor-augmented models. Different from IFEct which uses hard impute, MC uses soft impute to regularize the singular values when decomposing the residual matrix.
    * Only when latent factors (of unobservables) are strong and sparse, IFEct outperforms MC.
  - [Synthetic Controls] (case studies)

Identifying Assumptions:

1. **Function Form**: Additive separability of observables, unobservables, and idiosyncratic error term.
   - Hence, these models are scale dependent (Athey and Imbens, 2006) (e.g., log-transform outcome can invadiate this assumption).
2. **Strict Exogeneity**: Conditional on observables and unobservables, potential outcomes are independent of treatment assignment (i.e., baseline quasi-randomization)
   - In DiD, where unobservables = unit + time FEs, this assumption is the parallel trends assumption
3. **Low-dimensional Decomposition (Feasibility Assumption)**: Unobservable effects can be decomposed in low-dimension.
   - For the case that $U_{it} = f_t \times \lambda_i$ where f_t = common time trend (time FE), and λ_i = unit heterogeneity (unit FE). If $U_{it} = f_t \times \lambda_i$, DiD can satisfy this assumption. But this assumption is weaker than that of DID, and allows us to control for unobservables based on data.

Estimation Procedure:

1. Using all control observations, estimate the functions of both observable and unobservable variables (relying on Assumptions 1 and 3).
2. Predict the counterfactual outcomes for each treated unit using the obtained functions.
3. Calculate the difference in treatment effect for each treated individual.
4. By averaging over all treated individuals, you can obtain the Average Treatment Effect on the Treated (ATT).

Notes:

- Use jackknife when number of treated units is small (Liu et al., 2022, p.166).

29.9.3.2.1 Imputation Method Liu et al. (2022) can also account for treatment reversals and heterogeneous treatment effects.

Other imputation estimators include

- [@gardner2022two and @borusyak2021revisiting]
- Brown et al. (2023)

```
library(fect)

PanelMatch::dem

model.fect <-
    fect(
        Y = "y",
        D = "dem",
        X = "tradewb",
        data = na.omit(PanelMatch::dem),
        method = "fe",
        index = c("wbcode2", "year"),
        se = TRUE,
        parallel = TRUE,
        seed = 1234,
        # twfe
        force = "two-way"
    )
print(model.fect$est.avg)

plot(model.fect)

plot(model.fect, stats = "F.p")
```

F-test H_0: residual averages in the pre-treatment periods = 0

To see treatment reversal effects

```
plot(model.fect, stats = "F.p", type = 'exit')
```

29.9.3.2.2 Placebo Test By selecting a part of the data and excluding observations within a specified range to improve the model fitting, we then evaluate whether the estimated Average Treatment Effect (ATT) within this range significantly differs from zero. This approach helps us analyze the periods before treatment.

If this test fails, either the functional form or strict exogeneity assumption is problematic.

```
out.fect.p <-
    fect(
        Y = "y",
        D = "dem",
```

```
        X = "tradewb",
        data = na.omit(PanelMatch::dem),
        method = "fe",
        index = c("wbcode2", "year"),
        se = TRUE,
        placeboTest = TRUE,
        # using 3 periods
        placebo.period = c(-2, 0)
    )
plot(out.fect.p, proportion = 0.1, stats = "placebo.p")
```

29.9.3.2.3 (No) Carryover Effects Test The placebo test can be adapted to assess carryover effects by masking several post-treatment periods instead of pre-treatment ones. If no carryover effects are present, the average prediction error should approximate zero. For the carryover test, set `carryoverTest = TRUE`. Specify a post-treatment period range in carryover.period to exclude observations for model fitting, then evaluate if the estimated ATT significantly deviates from zero.

Even if we have carryover effects, in most cases of the staggered adoption setting, researchers are interested in the cumulative effects, or aggregated treatment effects, so it's okay.

```
out.fect.c <-
    fect(
        Y = "y",
        D = "dem",
        X = "tradewb",
        data = na.omit(PanelMatch::dem),
        method = "fe",
        index = c("wbcode2", "year"),
        se = TRUE,
        carryoverTest = TRUE,
        # how many periods of carryover
        carryover.period = c(1, 3)
    )
plot(out.fect.c,  stats = "carryover.p")
```

We have evidence of carryover effects.

29.9.3.3 Matrix Completion

Applications in marketing:

- Bronnenberg et al. (2020)

To estimate average causal effects in panel data with units exposed to treatment

intermittently, two literatures are pivotal:

- **Unconfoundedness** (Imbens and Rubin, 2015): Imputes missing potential control outcomes for treated units using observed outcomes from similar control units in previous periods.
- **Synthetic Control** (Abadie et al., 2010): Imputes missing control outcomes for treated units using weighted averages from control units, matching lagged outcomes between treated and control units.

Both exploit missing potential outcomes under different assumptions:

- Unconfoundedness assumes time patterns are stable across units.
- Synthetic control assumes unit patterns are stable over time.

Once regularization is applied, both approaches are applicable in similar settings (Athey et al., 2021).

Matrix Completion method, nesting both, is based on matrix factorization, focusing on imputing missing matrix elements assuming:

1. Complete matrix = low-rank matrix + noise.
2. Missingness is completely at random.

It's distinguished by not imposing factorization restrictions but utilizing regularization to define the estimator, particularly effective with the nuclear norm as a regularizer for complex missing patterns (Athey et al., 2021).

Contributions of Athey et al. (2021) matrix completion include:

1. Recognizing structured missing patterns allowing time correlation, enabling staggered adoption.
2. Modifying estimators for unregularized unit and time fixed effects.
3. Performing well across various T and N sizes, unlike unconfoundedness and synthetic control, which falter when $T >> N$ or $N >> T$, respectively.

Identifying Assumptions:

1. SUTVA: Potential outcomes indexed only by the unit's contemporaneous treatment.
2. No dynamic effects (it's okay under staggered adoption, it gives a different interpretation of estimand).

Setup:

- $Y_{it}(0)$ and $Y_{it}(1)$ represent potential outcomes of Y_{it}.
- W_{it} is a binary treatment indicator.

Aim to estimate the average effect for the treated:

$$\tau = \frac{\sum_{(i,t):W_{it}=1}[Y_{it}(1) - Y_{it}(0)]}{\sum_{i,t} W_{it}}$$

We observe all relevant values for $Y_{it}(1)$

We want to impute missing entries in the $Y(0)$ matrix for treated units with $W_{it} = 1$.

Define $\mathcal{M}$ as the set of pairs of indices (i,t), where $i \in N$ and $t \in T$, corresponding to missing entries with $W_{it} = 1$; $\mathcal{O}$ as the set of pairs of indices corresponding to observed entries in $Y(0)$ with $W_{it} = 0$.

Data is conceptualized as two $N \times T$ matrices, one incomplete and one complete:

$$Y = \begin{pmatrix} Y_{11} & Y_{12} & ? & \cdots & Y_{1T} \\ ? & ? & Y_{23} & \cdots & ? \\ Y_{31} & ? & Y_{33} & \cdots & ? \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ Y_{N1} & ? & Y_{N3} & \cdots & ? \end{pmatrix},$$

and

$$W = \begin{pmatrix} 0 & 0 & 1 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 1 \\ 0 & 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & 0 & \cdots & 1 \end{pmatrix},$$

where

$$W_{it} = \begin{cases} 1 & \text{if } (i,t) \in \mathcal{M}, \\ 0 & \text{if } (i,t) \in \mathcal{O}, \end{cases}$$

is an indicator for the event that the corresponding component of Y, that is Y_{it}, is missing.

Patterns of missing data in $\mathbf{Y}$:

- Block (treatment) structure with 2 special cases
  - Single-treated-period block structure (Imbens and Rubin, 2015)
  - Single-treated-unit block structure (Abadie et al., 2010)
- Staggered Adoption

Shape of matrix $\mathbf{Y}$:

- Thin ($N >> T$)
- Fat ($T >> N$)
- Square ($N \approx T$)

Combinations of patterns of missingness and shape create different literatures:

- Horizontal Regression = Thin matrix + single-treated-period block (focusing on cross-section correlation patterns)
- Vertical Regression = Fat matrix + single-treated-unit block (focusing on time-series correlation patterns)
- TWFE = Square matrix

To combine, we can exploit both stable patterns over time, and across units (e.g., TWFE, interactive FEs or matrix completion).

For the same factor model

$$\mathbf{Y} = \mathbf{U}\mathbf{V}^T +$$

where $\mathbf{U}$ is $N \times R$ and $\mathbf{V}$ is $T \times R$

The interactive FE literature focuses on a fixed number of factors R in $\mathbf{U}, \mathbf{V}$, while matrix completion focuses on impute $\mathbf{Y}$ using some forms regularization (e.g., nuclear norm).

- We can also estimate the number of factors R (Bai and Ng, 2002, Moon and Weidner (2015))

To use the nuclear norm minimization estimator, we must add a penalty term to regularize the objective function. However, before doing so, we need to explicitly estimate the time (λ_t) and unit (μ_i) fixed effects implicitly embedded in the missing data matrix to reduce the bias of the regularization term.

Specifically,

$$Y_{it} = L_{it} + \sum_{p=1}^{P}\sum_{q=1}^{Q} X_{ip}H_{pq}Z_{qt} + \mu_i + \lambda_t + V_{it}\beta + \epsilon_{it}$$

where

- X_{ip} is a matrix of p variables for unit i
- Z_{qt} is a matrix of q variables for time t
- V_{it} is a matrix of time-varying variables.

Lasso-type l_1 norm ($||H|| = \sum_{p=1}^{p}\sum_{q=1}^{Q}|H_{pq}|$) is used to shrink $H \to 0$

There are several options to regularize L:

1. Frobenius (i.e., Ridge): not informative since it imputes missing values as 0.
2. Nuclear Norm (i.e., Lasso): computationally feasible (using SOFT-IMPUTE algorithm (Mazumder et al., 2010)).
3. Rank (i.e., Subset selection): not computationally feasible

This method allows to

- use more covariates
- leverage data from treated units (can be used when treatment effect is constant and pattern of missing is not complex).
- have autocorrelated errors
- have weighted loss function (i.e., take into account the probability of outcomes for a unit being missing)

29.9.4 Gardner (2022) and Borusyak et al. (2021)

- Estimate the time and unit fixed effects separately
- Known as the imputation method (Borusyak et al., 2021) or two-stage DiD (Gardner, 2022)

```
# remotes::install_github("kylebutts/did2s")
library(did2s)
library(ggplot2)
library(fixest)
library(tidyverse)
data(base_stagg)


est <- did2s(
    data = base_stagg |> mutate(treat = if_else(time_to_treatment >= 0, 1, 0)),
    yname = "y",
    first_stage = ~ x1 | id + year,
    second_stage = ~ i(time_to_treatment, ref = c(-1,-1000)),
    treatment = "treat" ,
    cluster_var = "id"
)

fixest::esttable(est)
#>                                        est
#> Dependent Var.:                          y
#>
#> time_to_treatment = -9  0.3518** (0.1332)
#> time_to_treatment = -8  -0.3130* (0.1213)
#> time_to_treatment = -7    0.0894 (0.2367)
```

```
#> time_to_treatment = -6    0.0312 (0.2176)
#> time_to_treatment = -5   -0.2079 (0.1519)
#> time_to_treatment = -4   -0.1152 (0.1438)
#> time_to_treatment = -3   -0.0127 (0.1483)
#> time_to_treatment = -2    0.1503 (0.1440)
#> time_to_treatment = 0  -5.139*** (0.3680)
#> time_to_treatment = 1  -3.480*** (0.3784)
#> time_to_treatment = 2  -2.021*** (0.3055)
#> time_to_treatment = 3   -0.6965. (0.3947)
#> time_to_treatment = 4    1.070** (0.3501)
#> time_to_treatment = 5   2.173*** (0.4456)
#> time_to_treatment = 6   4.449*** (0.3680)
#> time_to_treatment = 7   4.864*** (0.3698)
#> time_to_treatment = 8   6.187*** (0.2702)
#> ---------------------- ------------------
#> S.E. type                          Custom
#> Observations                          950
#> R2                                0.62486
#> Adj. R2                           0.61843
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
fixest::iplot(
    est,
    main = "Event study",
    xlab = "Time to treatment",
    ref.line = -1
)
```

```
coefplot(est)
```

```
mult_est <- did2s::event_study(
    data = fixest::base_stagg |>
        dplyr::mutate(year_treated = dplyr::if_else(year_treated == 10000, 0, year_treated)),
    gname = "year_treated",
    idname = "id",
    tname = "year",
```

```
    yname = "y",
    estimator = "all"
)
#> Error in purrr::map(., function(y) { : i In index: 1.
#> i With name: y.
#> Caused by error in `.subset2()`:
#> ! no such index at level 1
did2s::plot_event_study(mult_est)
```

Borusyak et al. (2021) `didimputation`

This version is currently not working

```
library(didimputation)
library(fixest)
data("base_stagg")

did_imputation(
    data = base_stagg,
    yname = "y",
    gname = "year_treated",
    tname = "year",
    idname = "id"
)
```

29.9.5 De Chaisemartin and d'Haultfoeuille (2020)

use `twowayfeweights` from GitHub (De Chaisemartin and d'Haultfoeuille, 2020)

- Average instant treatment effect of changes in the treatment
  - This relaxes the no-carryover-effect assumption.
- Drawbacks:
  - Cannot observe treatment effects that manifest over time.

There still isn't a good package for this estimator.

```
# remotes::install_github("shuo-zhang-ucsb/did_multiplegt")
library(DIDmultiplegt)
library(fixest)
library(tidyverse)

data("base_stagg")

res <-
    did_multiplegt(
        df = base_stagg |>
            dplyr::mutate(treatment = dplyr::if_else(time_to_treatment < 0, 0, 1)),
        Y        = "y",
        G        = "year_treated",
        T        = "year",
        D        = "treatment",
        controls = "x1",
        # brep     = 20, # getting SE will take forever
        placebo  = 5,
        dynamic  = 5,
        average_effect = "simple"
    )

head(res)
#> $effect
#> treatment
#> -5.214207
#>
#> $N_effect
#> [1] 675
#>
#> $N_switchers_effect
#> [1] 45
#>
#> $dynamic_1
```

```
#> [1] -3.63556
#>
#> $N_dynamic_1
#> [1] 580
#>
#> $N_switchers_effect_1
#> [1] 40
```

I don't recommend the `TwoWayFEWeights` since it only gives the aggregated average treatment effect over all post-treatment periods, but not for each period.

```
library(TwoWayFEWeights)

res <- twowayfeweights(
    data = base_stagg |> dplyr::mutate(treatment = dplyr::if_else(time_to_treatment < 0
    Y = "y",
    G = "year_treated",
    T = "year",
    D = "treatment",
    summary_measures = T
)

print(res)
#> Under the common trends assumption, beta estimates a weighted sum of 45 ATTs.
#> 41 ATTs receive a positive weight, and 4 receive a negative weight.
#>
#> ------------------------------------------
#> Treat. var: treatment    ATTs    Σ weights
#> ------------------------------------------
#> Positive weights           41       1.0238
#> Negative weights            4      -0.0238
#> ------------------------------------------
#> Total                      45            1
#> ------------------------------------------
#>
#> Summary Measures:
#>   TWFE Coefficient (_fe): -3.4676
#>   min (Δ) compatible with _fe and Δ_TR = 0: 4.8357
#>   min (Δ) compatible with _fe and Δ_TR of a different sign: 36.1549
#>   Reference: Corollary 1, de Chaisemartin, C and D'Haultfoeuille, X (2020a)
#>
#> The development of this package was funded by the European Union (ERC, REALLYCREDIB
```

29.9.6 Callaway and Sant'Anna (2021)

- `staggered` package

- Group-time average treatment effect

```
library(staggered)
library(fixest)
data("base_stagg")

# simple weighted average
staggered(
    df = base_stagg,
    i = "id",
    t = "year",
    g = "year_treated",
    y = "y",
    estimand = "simple"
)
#>     estimate        se se_neyman
#> 1 -0.7110941 0.2211943 0.2214245

# cohort weighted average
staggered(
    df = base_stagg,
    i = "id",
    t = "year",
    g = "year_treated",
    y = "y",
    estimand = "cohort"
)
#>    estimate        se se_neyman
#> 1 -2.724242 0.2701093 0.2701745

# calendar weighted average
staggered(
    df = base_stagg,
    i = "id",
    t = "year",
    g = "year_treated",
    y = "y",
    estimand = "calendar"
)
#>     estimate        se se_neyman
#> 1 -0.5861831 0.1768297 0.1770729

res <- staggered(
    df = base_stagg,
    i = "id",
    t = "year",
```

```
    g = "year_treated",
    y = "y",
    estimand = "eventstudy",
    eventTime = -9:8
)
head(res)
#>      estimate        se se_neyman eventTime
#> 1  0.20418779 0.1045821 0.1045821        -9
#> 2 -0.06215104 0.1669703 0.1670886        -8
#> 3  0.02744671 0.1413273 0.1420377        -7
#> 4 -0.02131747 0.2203695 0.2206338        -6
#> 5 -0.30690897 0.2015697 0.2036412        -5
#> 6  0.05594029 0.1908101 0.1921745        -4


ggplot(
    res |> mutate(
        ymin_ptwise = estimate + 1.96 * se,
        ymax_ptwise = estimate - 1.96 * se
    ),
    aes(x = eventTime, y = estimate)
) +
    geom_pointrange(aes(ymin = ymin_ptwise, ymax = ymax_ptwise)) +
    geom_hline(yintercept = 0) +
    xlab("Event Time") +
    ylab("Estimate") +
    causalverse::ama_theme()
```

```
# Callaway and Sant'Anna estimator for the simple weighted average
staggered_cs(
    df = base_stagg,
    i = "id",
    t = "year",
    g = "year_treated",
    y = "y",
    estimand = "simple"
)
#>     estimate        se se_neyman
#> 1 -0.7994889 0.4484987 0.4486122

# Sun and Abraham estimator for the simple weighted average
staggered_sa(
    df = base_stagg,
    i = "id",
    t = "year",
    g = "year_treated",
    y = "y",
    estimand = "simple"
)
#>     estimate        se se_neyman
#> 1 -0.7551901 0.4407818 0.4409525
```

Fisher's Randomization Test (i.e., permutation test)

H_0: $TE = 0$

```r
staggered(
    df = base_stagg,
    i = "id",
    t = "year",
    g = "year_treated",
    y = "y",
    estimand = "simple",
    compute_fisher = T,
    num_fisher_permutations = 100
)
#>     estimate        se se_neyman fisher_pval fisher_pval_se_neyman
#> 1 -0.7110941 0.2211943 0.2214245           0                     0
#>   num_fisher_permutations
#> 1                     100
```

29.9.7 Sun and Abraham (2021)

This paper utilizes the Cohort Average Treatment Effects on the Treated (CATT), which measures the cohort-specific average difference in outcomes relative to those never treated, offering a more detailed analysis than Goodman-Bacon (2021). In scenarios lacking a never-treated group, this method designates the last cohort to be treated as the control group.

Parameter of interest is the cohort-specific ATT l periods from int ital treatment period e

$$CATT = E[Y_{i,e+I} - Y^{\infty}_{i,e+I} | E_i = e]$$

This paper uses an **interaction-weighted estimator** in a panel data setting, where the original paper Gibbons et al. (2018) used the same idea in a cross-sectional setting.

- Callaway and Sant'Anna (2021) explores group-time average treatment effects, employing cohorts that have not yet been treated as controls, and permits conditioning on time-varying covariates.
- Athey and Imbens (2022) examines the treatment effect in relation to the counterfactual outcome of the always-treated group, diverging from the conventional focus on the never-treated.
- Borusyak et al. (2021) presumes a uniform treatment effect across cohorts, effectively simplifying CATT to ATT.

Identifying Assumptions for dynamic TWFE:

1. **Parallel Trends**: Baseline outcomes follow parallel trends across cohorts before treatment.

- This gives us all CATT (including own, included bins, and excluded bins)

2. **No Anticipatory Behavior**: There is no effect of the treatment during pre-treatment periods, indicating that outcomes are not influenced by the anticipation of treatment.

3. **Treatment Effect Homogeneity**: The treatment effect is consistent across cohorts for each relative period. Each adoption cohort should have the same path of treatment effects. In other words, the trajectory of each treatment cohort is similar. Compare to other designs:

    1. Athey and Imbens (2022) assume heterogeneity of treatment effects vary over adoption cohorts, but not over time.

    2. Borusyak et al. (2021) assume heterogeneity of treatment effects vary over time, but not over adoption cohorts.

    3. Callaway and Sant'Anna (2021) assume heterogeneity of treatment effects vary over time and across cohorts.

    4. De Chaisemartin and D'haultfœuille (2023) assume heterogeneity of treatment effects vary across groups and over time.

    5. Goodman-Bacon (2021) assume heterogeneity either "vary across units but not over time" or "vary over time but not across units".

    6. Sun and Abraham (2021) allows for treatment effect heterogeneity across units and time.

Sources of Heterogeneous Treatment Effects

- Adoption cohorts can differ based on certain covariates. Similarly, composition of units within each adoption cohort is different.

- The response to treatment varies among cohorts if units self-select their initial treatment timing based on anticipated treatment effects. However, this self-selection is still compatible with the parallel trends assumption. This is true if units choose based on an evaluation of baseline outcomes - that is, if baseline outcomes are similar (following parallel trends), then we might not see selection into treatment based on the evaluation of the baseline outcome.

- Treatment effects can vary across cohorts due to calendar time-varying effects, such as changes in economic conditions.

Notes:

- If you do TWFE, you actually have to drop 2 terms to avoid multicollinearity:

    - Period right before treatment (this one was known before this paper)

  - Drop or bin or trim a distant lag period (this one was clarified by the paper). The reason is before of the multicollinearity in the linear relationship between TWFE and the relative period indicators.

- Contamination of the treatment effect estimates from excluded periods is a type of "normalization". To avoid this, we have to assume that all pre-treatment periods have the same CATT.
  - Sun and Abraham (2021) estimation method gives reasonable weights to CATT (i..e, weights that sum to 1, and are non negative). They estimate the weighted average of CATT where the weights are shares of cohorts that experience at least l periods after to treatment, normalized by the size of total periods g.

- Aggregation of CATT is similar to that of Callaway and Sant'Anna (2021)

Application

can use `fixest` in r with `sunab` function

```
library(fixest)
data("base_stagg")
res_sa20 = feols(y ~ x1 + sunab(year_treated, year) | id + year, base_stagg)
iplot(res_sa20)
```

```
summary(res_sa20, agg = "att")
#> OLS estimation, Dep. Var.: y
#> Observations: 950
#> Fixed-effects: id: 95,  year: 10
#> Standard-errors: Clustered (id)
```

```
#>      Estimate Std. Error  t value  Pr(>|t|)
#> x1   0.994678   0.018378 54.12293 < 2.2e-16 ***
#> ATT -1.133749   0.205070 -5.52858 2.882e-07 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> RMSE: 0.921817     Adj. R2: 0.887984
#>                  Within R2: 0.876406


summary(res_sa20, agg = c("att" = "year::[^-]"))
#> OLS estimation, Dep. Var.: y
#> Observations: 950
#> Fixed-effects: id: 95,  year: 10
#> Standard-errors: Clustered (id)
#>                       Estimate Std. Error   t value   Pr(>|t|)
#> x1                    0.994678   0.018378 54.122928  < 2.2e-16 ***
#> year::-9:cohort::10   0.351766   0.359073  0.979649 3.2977e-01
#> year::-8:cohort::9    0.033914   0.471437  0.071937 9.4281e-01
#> year::-8:cohort::10  -0.191932   0.352896 -0.543876 5.8781e-01
#> year::-7:cohort::8   -0.589387   0.736910 -0.799809 4.2584e-01
#> year::-7:cohort::9    0.872995   0.493427  1.769249 8.0096e-02 .
#> year::-7:cohort::10   0.019512   0.603411  0.032336 9.7427e-01
#> year::-6:cohort::7   -0.042147   0.865736 -0.048683 9.6127e-01
#> year::-6:cohort::8   -0.657571   0.573257 -1.147078 2.5426e-01
#> year::-6:cohort::9    0.877743   0.533331  1.645775 1.0315e-01
#> year::-6:cohort::10  -0.403635   0.347412 -1.161832 2.4825e-01
#> year::-5:cohort::6   -0.658034   0.913407 -0.720418 4.7306e-01
#> year::-5:cohort::7   -0.316974   0.697939 -0.454158 6.5076e-01
#> year::-5:cohort::8   -0.238213   0.469744 -0.507113 6.1326e-01
#> year::-5:cohort::9    0.301477   0.604201  0.498968 6.1897e-01
#> year::-5:cohort::10  -0.564801   0.463214 -1.219308 2.2578e-01
#> year::-4:cohort::5   -0.983453   0.634492 -1.549984 1.2451e-01
#> year::-4:cohort::6    0.360407   0.858316  0.419900 6.7552e-01
#> year::-4:cohort::7   -0.430610   0.661356 -0.651102 5.1657e-01
#> year::-4:cohort::8   -0.895195   0.374901 -2.387816 1.8949e-02 *
#> year::-4:cohort::9   -0.392478   0.439547 -0.892914 3.7418e-01
#> year::-4:cohort::10   0.519001   0.597880  0.868069 3.8757e-01
#> year::-3:cohort::4    0.591288   0.680169  0.869324 3.8688e-01
#> year::-3:cohort::5   -1.000650   0.971741 -1.029749 3.0577e-01
#> year::-3:cohort::6    0.072188   0.652641  0.110609 9.1216e-01
#> year::-3:cohort::7   -0.836820   0.804275 -1.040465 3.0079e-01
#> year::-3:cohort::8   -0.783148   0.701312 -1.116691 2.6697e-01
#> year::-3:cohort::9    0.811285   0.564470  1.437251 1.5397e-01
#> year::-3:cohort::10   0.527203   0.320051  1.647250 1.0285e-01
#> year::-2:cohort::3    0.036941   0.673771  0.054828 9.5639e-01
```

```
#> year::-2:cohort::4   0.832250   0.859544  0.968246 3.3541e-01
#> year::-2:cohort::5  -1.574086   0.525563 -2.995051 3.5076e-03 **
#> year::-2:cohort::6   0.311758   0.832095  0.374666 7.0875e-01
#> year::-2:cohort::7  -0.558631   0.871993 -0.640638 5.2332e-01
#> year::-2:cohort::8   0.429591   0.305270  1.407250 1.6265e-01
#> year::-2:cohort::9   1.201899   0.819186  1.467188 1.4566e-01
#> year::-2:cohort::10 -0.002429   0.682087 -0.003562 9.9717e-01
#> att                 -1.133749   0.205070 -5.528584 2.8820e-07 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> RMSE: 0.921817     Adj. R2: 0.887984
#>                  Within R2: 0.876406

# alternatively
summary(res_sa20, agg = c("att" = "year::[012345678]")) |>
    etable(digits = 2)
#>                           summary(res_..
#> Dependent Var.:                        y
#>
#> x1                        0.99*** (0.02)
#> year = -9 x cohort = 10    0.35 (0.36)
#> year = -8 x cohort = 9     0.03 (0.47)
#> year = -8 x cohort = 10   -0.19 (0.35)
#> year = -7 x cohort = 8    -0.59 (0.74)
#> year = -7 x cohort = 9    0.87. (0.49)
#> year = -7 x cohort = 10    0.02 (0.60)
#> year = -6 x cohort = 7    -0.04 (0.87)
#> year = -6 x cohort = 8    -0.66 (0.57)
#> year = -6 x cohort = 9     0.88 (0.53)
#> year = -6 x cohort = 10   -0.40 (0.35)
#> year = -5 x cohort = 6    -0.66 (0.91)
#> year = -5 x cohort = 7    -0.32 (0.70)
#> year = -5 x cohort = 8    -0.24 (0.47)
#> year = -5 x cohort = 9     0.30 (0.60)
#> year = -5 x cohort = 10   -0.56 (0.46)
#> year = -4 x cohort = 5    -0.98 (0.63)
#> year = -4 x cohort = 6     0.36 (0.86)
#> year = -4 x cohort = 7    -0.43 (0.66)
#> year = -4 x cohort = 8   -0.90* (0.37)
#> year = -4 x cohort = 9    -0.39 (0.44)
#> year = -4 x cohort = 10    0.52 (0.60)
#> year = -3 x cohort = 4     0.59 (0.68)
#> year = -3 x cohort = 5     -1.0 (0.97)
#> year = -3 x cohort = 6     0.07 (0.65)
#> year = -3 x cohort = 7    -0.84 (0.80)
```

```
#> year = -3 x cohort = 8    -0.78 (0.70)
#> year = -3 x cohort = 9     0.81 (0.56)
#> year = -3 x cohort = 10    0.53 (0.32)
#> year = -2 x cohort = 3     0.04 (0.67)
#> year = -2 x cohort = 4     0.83 (0.86)
#> year = -2 x cohort = 5   -1.6** (0.53)
#> year = -2 x cohort = 6     0.31 (0.83)
#> year = -2 x cohort = 7    -0.56 (0.87)
#> year = -2 x cohort = 8     0.43 (0.31)
#> year = -2 x cohort = 9      1.2 (0.82)
#> year = -2 x cohort = 10  -0.002 (0.68)
#> att                     -1.1*** (0.21)
#> Fixed-Effects:          --------------
#> id                                Yes
#> year                              Yes
#> ----------------------- --------------
#> S.E.: Clustered                 by: id
#> Observations                       950
#> R2                             0.90982
#> Within R2                      0.87641
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Using the same syntax as `fixest`

```r
# devtools::install_github("kylebutts/fwlplot")
library(fwlplot)
fwl_plot(y ~ x1, data = base_stagg)
```

```
fwl_plot(y ~ x1 | id + year, data = base_stagg, n_sample = 100)
```

```
fwl_plot(y ~ x1 | id + year, data = base_stagg, n_sample = 100, fsplit = ~ treated)
```

29.9.8 Wooldridge (2022)

use etwfe(Extended two-way Fixed Effects) (Wooldridge, 2022)

29.9.9 Doubly Robust DiD

Also known as the locally efficient doubly robust DiD (Sant'Anna and Zhao, 2020)

Code example by the authors

The package (not method) is rather limited application:

- Use OLS (cannot handle `glm`)
- Canonical DiD only (cannot handle DDD).

```
library(DRDID)
data("nsw_long")
eval_lalonde_cps <-
    subset(nsw_long, nsw_long$treated == 0 | nsw_long$sample == 2)
head(eval_lalonde_cps)
#>   id year treated age educ black married nodegree dwincl      re74 hisp
#> 1  1 1975      NA  42   16     0       1        0     NA     0.000    0
#> 2  1 1978      NA  42   16     0       1        0     NA     0.000    0
#> 3  2 1975      NA  20   13     0       0        0     NA  2366.794    0
#> 4  2 1978      NA  20   13     0       0        0     NA  2366.794    0
#> 5  3 1975      NA  37   12     0       1        0     NA 25862.322    0
#> 6  3 1978      NA  37   12     0       1        0     NA 25862.322    0
```

```
#>   early_ra sample experimental         re
#> 1       NA      2            0     0.0000
#> 2       NA      2            0   100.4854
#> 3       NA      2            0  3317.4678
#> 4       NA      2            0  4793.7451
#> 5       NA      2            0 22781.8555
#> 6       NA      2            0 25564.6699


# locally efficient doubly robust DiD Estimators for the ATT
out <-
    drdid(
        yname = "re",
        tname = "year",
        idname = "id",
        dname = "experimental",
        xformla = ~ age + educ + black + married + nodegree + hisp + re74,
        data = eval_lalonde_cps,
        panel = TRUE
    )
summary(out)
#>  Call:
#> drdid(yname = "re", tname = "year", idname = "id", dname = "experimental",
#>     xformla = ~age + educ + black + married + nodegree + hisp +
#>         re74, data = eval_lalonde_cps, panel = TRUE)
#> ------------------------------------------------------------------
#>  Further improved locally efficient DR DID estimator for the ATT:
#>
#>    ATT     Std. Error  t value    Pr(>|t|)  [95% Conf. Interval]
#> -901.2703   393.6247   -2.2897     0.022    -1672.7747  -129.766
#> ------------------------------------------------------------------
#>  Estimator based on panel data.
#>  Outcome regression est. method: weighted least squares.
#>  Propensity score est. method: inverse prob. tilting.
#>  Analytical standard error.
#> ------------------------------------------------------------------
#>  See Sant'Anna and Zhao (2020) for details.



# Improved locally efficient doubly robust DiD estimator
# for the ATT, with panel data
# drdid_imp_panel()

# Locally efficient doubly robust DiD estimator for the ATT,
```

```
# with panel data
# drdid_panel()

# Locally efficient doubly robust DiD estimator for the ATT,
# with repeated cross-section data
# drdid_rc()

# Improved locally efficient doubly robust DiD estimator for the ATT,
# with repeated cross-section data
# drdid_imp_rc()
```

29.9.10 Augmented/Forward DID

- DID Methods for Limited Pre-Treatment Periods:

Method	Scenario	Approach
Augmented DID (Li and Van den Bulte, 2023)	Treatment outcome is outside the range of control units	Constructs the treatment counterfactual using a scaled average of control units
Forward DID (Li, 2024)	Treatment outcome is within the range of control units	Uses a forward selection algorithm to choose relevant control units before applying DID

29.10 Multiple Treatments

When you have 2 treatments in a setting, you should always try to model both of them under one regression to see whether they are significantly different.

- Never use one treated groups as control for the other, and run separate regression.
- Could check this answer

$$\begin{aligned} Y_{it} &= \alpha + \gamma_1 Treat1_i + \gamma_2 Treat2_i + \lambda Post_t \\ &\quad + \delta_1(Treat1_i \times Post_t) + \delta_2(Treat2_i \times Post_t) + \epsilon_{it} \end{aligned}$$

(Fricke, 2017)

(De Chaisemartin and D'haultfœuille, 2023) video code

29.11 Mediation Under DiD

Check this post

29.12 Assumptions

- **Parallel Trends**: Difference between the treatment and control groups remain constant if there were no treatment.
  - should be used in cases where
    - you observe before and after an event
    - you have treatment and control groups
  - not in cases where
    - treatment is not random
    - confounders.
  - To support we use
    - Placebo test
    - Prior Parallel Trends Test
- **Linear additive effects** (of group/unit specific and time-specific):
  - If they are not additively interact, we have to use the weighted 2FE estimator (Imai and Kim, 2021)
  - Typically seen in the Staggered Dif-n-dif
- No anticipation: There is no causal effect of the treatment before its implementation.

Possible issues

- Estimate dependent on functional form:
  - When the size of the response depends (nonlinearly) on the size of the intervention, we might want to look at the the difference in the group with high intensity vs. low.
- Selection on (time–varying) unobservables
  - Can use the overall sensitivity of coefficient estimates to hidden bias using Rosenbaum Bounds
- Long-term effects
  - Parallel trends are more likely to be observed over shorter period (window of observation)
- Heterogeneous effects

  - Different intensity (e.g., doses) for different groups.

- Ashenfelter dip (Ashenfelter and Card, 1985) (job training program participant are more likely to experience an earning drop prior enrolling in these programs)
  - Participants are systemically different from nonparticipants before the treatment, leading to the question of permanent or transitory changes.
  - A fix to this transient endogeneity is to calculate long-run differences (exclude a number of periods symmetrically around the adoption/ implementation date). If we see a sustained impact, then we have strong evidence for the causal impact of a policy. (Proserpio and Zervas, 2017a) (Heckman and Smith, 1999) (Jepsen et al., 2014) (Li et al., 2011)

- Response to event might not be immediate (can't be observed right away in the dependent variable)
  - Using lagged dependent variable Y_{it-1} might be more appropriate (Blundell and Bond, 1998)

- Other factors that affect the difference in trends between the two groups (i.e., treatment and control) will bias your estimation.

- Correlated observations within a group or time

- Incidental parameters problems (Lancaster, 2000): it's always better to use individual and time fixed effect.

- When examining the effects of variation in treatment timing, we have to be careful because negative weights (per group) can be negative if there is a heterogeneity in the treatment effects over time. Example: [Athey and Imbens (2022)](Borusyak et al., 2021)(Goodman-Bacon, 2021). In this case you should use new estimands proposed by @callaway2021difference(De Chaisemartin and d'Haultfoeuille, 2020), in the `did` package. If you expect lags and leads, see (Sun and Abraham, 2021)

- (Gibbons et al., 2018) caution when we suspect the treatment effect and treatment variance vary across groups

29.12.1 Prior Parallel Trends Test

1. Plot the average outcomes over time for both treatment and control group before and after the treatment in time.
2. Statistical test for difference in trends (**using data from before the treatment period**)

$$Y = \alpha_g + \beta_1 T + \beta_2 T \times G + \epsilon$$

where

- Y = the outcome variable
- α_g = group fixed effects
- T = time (e.g., specific year, or month)
- β_2 = different time trends for each group

Hence, if $\beta_2 = 0$ provides evidence that there are no differences in the trend for the two groups prior the time treatment.

You can also use different functional forms (e..g, polynomial or nonlinear).

If $\beta_2 \neq 0$ statistically, possible reasons can be:

- Statistical significance can be driven by large sample
- Or the trends are so consistent, and just one period deviation can throw off the trends. Hence, statistical statistical significance.

Technically, we can still salvage the research by including time fixed effects, instead of just the before-and-after time fixed effect (actually, most researchers do this mechanically anyway nowadays). However, a side effect can be that the time fixed effects can also absorb some part your treatment effect as well, especially in cases where the treatment effects vary with time (i.e., stronger or weaker over time) (Wolfers, 2003).

Debate:

- (Kahn-Lang and Lang, 2020) argue that DiD will be more plausible when the treatment and control groups are similar not only in **trends**, but also in **levels**. Because when we observe dissimilar in levels prior to the treatment, why is it okay to think that this will not affect future trends?
  - Show a plot of the dependent variable's time series for treated and control groups and also a similar plot with matched sample. (Ryan et al., 2019) show evidence of matched DiD did well in the setting of non-parallel trends (at least in health care setting).
  - In the case that we don't have similar levels ex ante between treatment and control groups, functional form assumptions matter and we need justification for our choice.
- Pre-trend statistical tests: (Roth, 2022) provides evidence that these test are usually under powered.
  - See PretrendsPower and pretrends packages for correcting this.
- Parallel trends assumption is specific to both the transformation and units of the outcome (Roth and Sant'Anna, 2023)
  - See falsification test (H_0: parallel trends is insensitive to functional form).

```r
library(tidyverse)
library(fixest)
od <- causaldata::organ_donations %>%
    # Use only pre-treatment data
    filter(Quarter_Num <= 3) %>%
    # Treatment variable
    dplyr::mutate(California = State == 'California')

# use my package
causalverse::plot_par_trends(
    data = od,
    metrics_and_names = list("Rate" = "Rate"),
    treatment_status_var = "California",
    time_var = list(Quarter_Num = "Time"),
    display_CI = F
)
#> [[1]]
```

```r
# do it manually
# always good but plot the dependent out
od |>
    # group by treatment status and time
    dplyr::group_by(California, Quarter) |>
    dplyr::summarize_all(mean) |>
    dplyr::ungroup() |>
    # view()
```

```
ggplot2::ggplot(aes(x = Quarter_Num, y = Rate, color = California)) +
ggplot2::geom_line() +
causalverse::ama_theme()
```

```
# but it's also important to use statistical test
prior_trend <- fixest::feols(Rate ~ i(Quarter_Num, California) | State + Quarter,
               data = od)

fixest::coefplot(prior_trend, grid = F)
```

```
fixest::iplot(prior_trend, grid = F)
```

This is alarming since one of the periods is significantly different from 0, which means that our parallel trends assumption is not plausible.

In cases where you might have violations of parallel trends assumption, check (Rambachan and Roth, 2023)

- Impose restrictions on how different the post-treatment violations of parallel trends can be from the pre-trends.

- Partial identification of causal parameter
- Sensitivity analysis

```
# https://github.com/asheshrambachan/HonestDiD
# remotes::install_github("asheshrambachan/HonestDiD")
# library(HonestDiD)
```

Alternatively, Ban and Kedagni (2022) propose a method that with an information set (i.e., pre-treatment covariates), and an assumption on the selection bias in the post-treatment period (i.e., lies within the convex hull of all selection biases), they can still identify a set of ATT, and with stricter assumption on selection bias from the policymakers perspective, they can also have a point estimate.

Alternatively, we can use the `pretrends` package to examine our assumptions (Roth, 2022)

29.12.2 Placebo Test

Procedure:

1. Sample data only in the period before the treatment in time.
2. Consider different fake cutoff in time, either
    1. Try the whole sequence in time
    2. Generate random treatment period, and use **randomization inference** to account for sampling distribution of the fake effect.
3. Estimate the DiD model but with the post-time = 1 with the fake cutoff
4. A significant DiD coefficient means that you violate the parallel trends! You have a big problem.

Alternatively,

- When data have multiple control groups, drop the treated group, and assign another control group as a "fake" treated group. But even if it fails (i.e., you find a significant DiD effect) among the control groups, it can still be fine. However, this method is used under Synthetic Control

Code by theeffectbook.net

```
library(tidyverse)
library(fixest)

od <- causaldata::organ_donations %>%
    # Use only pre-treatment data
    dplyr::filter(Quarter_Num <= 3) %>%

    # Create fake treatment variables
    dplyr::mutate(
        FakeTreat1 = State == 'California' &
```

```
            Quarter %in% c('Q12011', 'Q22011'),
        FakeTreat2 = State == 'California' &
            Quarter == 'Q22011'
    )


clfe1 <- fixest::feols(Rate ~ FakeTreat1 | State + Quarter,
               data = od)
clfe2 <- fixest::feols(Rate ~ FakeTreat2 | State + Quarter,
               data = od)

fixest::etable(clfe1,clfe2)
#>                            clfe1            clfe2
#> Dependent Var.:             Rate             Rate
#>
#> FakeTreat1TRUE  0.0061 (0.0051)
#> FakeTreat2TRUE                  -0.0017 (0.0028)
#> Fixed-Effects:  --------------- ----------------
#> State                        Yes              Yes
#> Quarter                      Yes              Yes
#> _______________ _______________ ________________
#> S.E.: Clustered       by: State        by: State
#> Observations                 81               81
#> R2                      0.99377          0.99376
#> Within R2               0.00192          0.00015
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We would like the "supposed" DiD to be insignificant.

29.12.3 Assumption Violations

1. Endogenous Timing

If the timing of units can be influenced by strategic decisions in a DID analysis, an instrumental variable approach with a control function can be used to control for endogeneity in timing.

2. Questionable Counterfactuals

In situations where the control units may not serve as a reliable counterfactual for the treated units, matching methods such as propensity score matching or generalized random forest can be utilized. Additional methods can be found in Matching Methods.

29.12.4 Robustness Checks

- Placebo DiD (if the DiD estimate $\neq 0$, parallel trend is violated, and original DiD is biased):
  - Group: Use fake treatment groups: A population that was **not** affect by the treatment
  - Time: Redo the DiD analysis for period before the treatment (expected treatment effect is 0) (e.g., for previous year or period).
- Possible alternative control group: Expected results should be similar
- Try different windows (further away from the treatment point, other factors can creep in and nullify your effect).
- Treatment Reversal (what if we don't see the treatment event)
- Higher-order polynomial time trend (to relax linearity assumption)
- Test whether other dependent variables that should not be affected by the event are indeed unaffected.
  - Use the same control and treatment period (DiD $\neq 0$, there is a problem)
- The **triple-difference strategy** involves examining the interaction between the **treatment variable** and **the probability of being affected by the program**, and the group-level participation rate. The identification assumption is that there are no differential trends between high and low participation groups in early versus late implementing countries.

Chapter 30

Changes-in-Changes

- **Introduction**
  - The Changes-in-Changes (CiC) estimator, introduced by Athey and Imbens (2006), is an alternative to the Difference-in-Differences (DiD) strategy.
  - Unlike traditional DiD, which estimates the Average Treatment Effect on the Treated (ATT), CiC focuses on the Quantile Treatment Effect on the Treated (QTT).
  - QTT captures the difference between potential outcome distributions for treated units at a specific quantile.
  - **Beyond Averages:** Policymakers often look beyond average program impacts, considering how benefits are distributed across different groups.
    - **Job Training Example:** Two programs with the same negative average impact may be treated differently: one benefiting high earners might be rejected, while one benefiting low earners could be approved.
    - **Traditional Methods' Limitations:** Methods like linear regression, which assume uniform effects, fail to capture important distributional differences.
    - **QTEs' Advantage:** QTE methods are tailored for analyzing how treatment effects vary across different segments of a population.
  - **QTE vs. ATE:** While QTEs provide detailed insights into distributional impacts, they also allow for the recovery of ATEs. However, ATEs are usually identified under weaker assumptions, making QTEs more suitable for exploring the shape of treatment effects rather than just their central tendency.
- **Key Concepts**

  - **Quantile Treatment Effect on the Treated (QTT):** Difference in quantiles of treated units' potential outcome distributions.
  - **Rank Preservation:** Assumes each unit's rank remains constant across potential outcome distributions—this is a strong assumption.
  - **Counterfactual Distribution:** Estimation focuses on determining this distribution for the treated units in period 1.

- **Estimating QTT**
  - CiC uses four distributions from a 2x2 DiD design:
    1. $F_{Y(0),00}$: CDF of $Y(0)$ for control units in period 0.
    2. $F_{Y(0),10}$: CDF of $Y(0)$ for treatment units in period 0.
    3. $F_{Y(0),01}$: CDF of $Y(0)$ for control units in period 1.
    4. $F_{Y(1),11}$: CDF of $Y(1)$ for treatment units in period 1.
  - QTT is defined as the difference between the inverses of $F_{Y(1),11}$ and the counterfactual distribution $F_{Y(0),11}$ at quantile q:

$$\Delta_\theta^{QTT} = F_{Y(1),11}^{-1}(\theta) - F_{Y(0),11}^{-1}(\theta)$$

- **Estimation Process**
  - **Counterfactual CDF:**

$$\hat{F}_{Y(0),11}(y) = F_{y,01}\left(F_{y,00}^{-1}\left(F_{y,10}(y)\right)\right)$$

  - **Equivalent Expression:**

$$\hat{F}_{Y(0),11}^{-1}(\theta) = F_{y,01}^{-1}\left(F_{y,00}\left(F_{y,10}^{-1}(\theta)\right)\right)$$

  - **Treatment Effect Estimate:**

$$\hat{\Delta}_\theta^{CIC} = F_{Y(1),11}^{-1}(\theta) - \hat{F}_{Y(0),11}^{-1}(\theta)$$

  - **Equivalently:**

  Δ_θ^{CIC} is the difference between two QTE estimates:

$$\Delta_\theta^{CIC} = \Delta_{\theta,1}^{QTE} - \Delta_{\theta',0}^{QTE}$$

  where:
  - $\Delta_{\theta,1}^{QTT}$ = change over time in y at quantile θ for $D = 1$ group.
  - $\Delta_{\theta',0}^{QTU}$ = change over time in y at quantile θ' for $D = 0$ group, where q' is the quantile in the $D = 0, T = 0$ distribution corresponding to the value of y associated with quantile θ in the $D = 1, T = 0$ distribution.

- **Marketing Example**

  - Suppose a company implements a new online marketing strategy aimed at improving customer retention rates.
  - **QTT:** The goal is to estimate the effect of the strategy on customer retention rates at different quantiles (e.g., median retention rate).
  - **Rank Preservation:** Assumes customers' rank in retention distribution remains the same, regardless of the strategy—this assumption is strong and should be carefully considered.
  - **Counterfactual:** CiC helps estimate how retention rates would have changed without the new strategy by comparing it with a control group.

- **References**
  - Athey and Imbens (2006)
  - Frölich and Melly (2013): IV-based
  - Callaway and Li (2019): panel data
  - Huber et al. (2022)

- **Additional Resources**
  - Code examples available in Stata.

30.1 Application

30.1.1 ECIC package

```
library(ecic)
data(dat, package = "ecic")
mod =
  ecic(
    yvar  = lemp,          # dependent variable
    gvar  = first.treat,   # group indicator
    tvar  = year,          # time indicator
    ivar  = countyreal,    # unit ID
    dat   = dat,           # dataset
    boot  = "weighted",    # bootstrap proceduce ("no", "normal", or "weighted")
    nReps = 3             # number of bootstrap runs
    )
mod_res <- summary(mod)
mod_res
#>   perc    coefs          se
#> 1  0.1 1.206140 0.021351711
#> 2  0.2 1.316599 0.009225026
#> 3  0.3 1.449963 0.001859468
#> 4  0.4 1.583415 0.015296156
#> 5  0.5 1.739932 0.011240454
#> 6  0.6 1.915558 0.013060348
```

```
#> 7  0.7 2.114966 0.014482208
#> 8  0.8 2.363105 0.005173865
#> 9  0.9 2.779202 0.020831180

ecic_plot(mod_res)
```

30.1.2 QTE package

```
library(qte)
data(lalonde)

# randomized setting
# qte is identical to qtet
jt.rand <-
    ci.qtet(
        re78 ~ treat,
        data = lalonde.exp,
        iters = 10
    )
summary(jt.rand)
#>
#> Quantile Treatment Effect:
#>
#> tau  QTE Std. Error
#> 0.05    0.00    0.00
#> 0.1     0.00    0.00
```

```
#> 0.15    0.00    0.00
#> 0.2     0.00    0.00
#> 0.25  338.65  281.41
#> 0.3   846.40  342.60
#> 0.35 1451.51  414.82
#> 0.4  1177.72  626.79
#> 0.45 1396.08  824.15
#> 0.5  1123.55  531.67
#> 0.55 1181.54  343.91
#> 0.6  1466.51  453.68
#> 0.65 2115.04  672.77
#> 0.7  1795.12  808.87
#> 0.75 2347.49  819.03
#> 0.8  2278.12  828.77
#> 0.85 2178.28 1041.09
#> 0.9  3239.60 1624.72
#> 0.95 3979.62 2843.35
#>
#> Average Treatment Effect:    1794.34
#>   Std. Error:        566.19
ggqte(jt.rand)
```

```
# conditional independence assumption (CIA)
jt.cia <- ci.qte(
    re78 ~ treat,
    xformla =  ~ age + education,
    data = lalonde.psid,
```

```
    iters = 10
)
summary(jt.cia)
#>
#> Quantile Treatment Effect:
#>
#> tau  QTE Std. Error
#> 0.05      0.00        0.00
#> 0.1       0.00        0.00
#> 0.15  -4433.18     1202.23
#> 0.2   -8219.15      833.89
#> 0.25 -10435.74      957.34
#> 0.3  -12232.03     1277.29
#> 0.35 -12428.30     1521.99
#> 0.4  -14195.24     2331.68
#> 0.45 -14248.66     3012.78
#> 0.5  -15538.67     3195.32
#> 0.55 -16550.71     3340.80
#> 0.6  -15595.02     3215.55
#> 0.65 -15827.52     3308.46
#> 0.7  -16090.32     4273.72
#> 0.75 -16091.49     4771.84
#> 0.8  -17864.76     4311.49
#> 0.85 -16756.71     4083.31
#> 0.9  -17914.99     4245.21
#> 0.95 -23646.22     3310.14
#>
#> Average Treatment Effect:    -13435.40
#>   Std. Error:        2130.66
ggqte(jt.cia)
```

```
jt.ciat <- ci.qtet(
    re78 ~ treat,
    xformla =  ~ age + education,
    data = lalonde.psid,
    iters = 10
)
summary(jt.ciat)
#>
#> Quantile Treatment Effect:
#>
#> tau  QTE Std. Error
#> 0.05      0.00        0.00
#> 0.1   -1018.15      603.16
#> 0.15  -3251.00     1293.89
#> 0.2   -7240.86      687.09
#> 0.25  -8379.94      419.72
#> 0.3   -8758.82      481.41
#> 0.35  -9897.44      518.42
#> 0.4  -10239.57      413.92
#> 0.45 -10751.39      561.51
#> 0.5  -10570.14      527.01
#> 0.55 -11348.96      657.51
#> 0.6  -11550.84      755.33
#> 0.65 -12203.56      769.28
#> 0.7  -13277.72     1041.18
#> 0.75 -14011.74     1253.29
```

```
#> 0.8  -14373.95     1104.17
#> 0.85 -14499.18     1282.32
#> 0.9  -15008.63     1776.53
#> 0.95 -15954.05     2406.83
#>
#> Average Treatment Effect:    4266.19
#>   Std. Error:        484.33
ggqte(jt.ciat)
```

- **QTE** compares quantiles of the entire population under treatment and control, whereas **QTET** compares quantiles within the treated group itself. This difference means that QTE reflects the overall population-level impact, while QTET focuses on the treated group's specific impact.
- **CIA** enables identification of both QTE and QTET, but since QTET is conditional on treatment, it might reflect different effects than QTE, especially when the treatment effect is heterogeneous across different subpopulations. For example, the QTE could show a more generalized effect across all individuals, while the QTET may reveal stronger or weaker effects for the subgroup that actually received the treatment.

These are DID-like models

1. With the distributional difference-in-differences assumption (Fan and Yu, 2012, Callaway and Li (2019)), which is an extension of the parallel trends assumption, we can estimate QTET.

```
# distributional DiD assumption
jt.pqtet <- panel.qtet(
```

```r
    re ~ treat,
    t = 1978,
    tmin1 = 1975,
    tmin2 = 1974,
    tname = "year",
    idname = "id",
    data = lalonde.psid.panel,
    iters = 10
)
summary(jt.pqtet)
#>
#> Quantile Treatment Effect:
#>
#> tau  QTE Std. Error
#> 0.05  4779.21     1357.47
#> 0.1   1987.35      479.31
#> 0.15   842.95     2840.61
#> 0.2  -7366.04     3899.61
#> 0.25 -8449.96     1312.56
#> 0.3  -7992.15     1338.14
#> 0.35 -7429.21     1292.52
#> 0.4  -6597.37     1451.63
#> 0.45 -5519.45     1558.49
#> 0.5  -4702.88     1341.16
#> 0.55 -3904.52     1357.87
#> 0.6  -2741.80     1518.28
#> 0.65 -1507.31     1330.27
#> 0.7   -771.12     1565.81
#> 0.75   707.81     1140.12
#> 0.8    580.00      927.67
#> 0.85   821.75     1971.49
#> 0.9   -250.77     1647.18
#> 0.95 -1874.54     2562.85
#>
#> Average Treatment Effect:    2326.51
#>   Std. Error:        848.03
ggqte(jt.pqtet)
```

2. With 2 periods, the distributional DiD assumption can partially identify QTET with bounds (Fan and Yu, 2012)

```
res_bound <-
    bounds(
        re ~ treat,
        t = 1978,
        tmin1 = 1975,
        data = lalonde.psid.panel,
        idname = "id",
        tname = "year"
    )
summary(res_bound)
#>
#> Bounds on the Quantile Treatment Effect on the Treated:
#>
#> tau  Lower Bound Upper Bound
#>          tau  Lower Bound Upper Bound
#>         0.05       -51.72           0
#>          0.1     -1220.84           0
#>         0.15      -1881.9           0
#>          0.2     -2601.32           0
#>         0.25     -2916.38      485.23
#>          0.3     -3080.16      943.05
#>         0.35     -3327.89     1505.98
#>          0.4     -3240.59     2133.59
#>         0.45     -2982.51     2616.84
```

```
#>          0.5     -3108.01      2566.2
#>         0.55     -3342.66     2672.82
#>          0.6      -3491.4      3065.7
#>         0.65     -3739.74     3349.74
#>          0.7     -4647.82     2992.03
#>         0.75     -4826.78     3219.32
#>          0.8      -5801.7     2702.33
#>         0.85     -6588.61     2499.41
#>          0.9     -8953.84     2020.84
#>         0.95    -14283.61      397.04
#>
#> Average Treatment Effect on the Treated: 2326.51
plot(res_bound)
```

3. With a restrictive assumption that difference in the quantiles of the distribution of potential outcomes for the treated and untreated groups be the same for all values of quantiles, we can have the mean DiD model

```
jt.mdid <- ddid2(
    re ~ treat,
    t = 1978,
    tmin1 = 1975,
    tname = "year",
    idname = "id",
    data = lalonde.psid.panel,
    iters = 10
)
```

```r
summary(jt.mdid)
#>
#> Quantile Treatment Effect:
#>
#> tau  QTE Std. Error
#> 0.05 10616.61      768.05
#> 0.1   5019.83      434.31
#> 0.15  2388.12      235.74
#> 0.2   1033.23      227.81
#> 0.25   485.23      432.95
#> 0.3    943.05      519.98
#> 0.35   931.45      519.38
#> 0.4    945.35      722.58
#> 0.45  1205.88      792.34
#> 0.5   1362.11      968.32
#> 0.55  1279.05     1085.89
#> 0.6   1618.13     1026.47
#> 0.65  1834.30     1096.94
#> 0.7   1326.06     1053.55
#> 0.75  1586.35     1181.80
#> 0.8   1256.09     1023.67
#> 0.85   723.10     1865.37
#> 0.9    251.36     2418.52
#> 0.95 -1509.92     1467.98
#>
#> Average Treatment Effect:    2326.51
#>   Std. Error:        672.86
plot(jt.mdid)
```

On top of the distributional DiD assumption, we need **copula stability** assumption (i.e., If, before the treatment, the units with the highest outcomes were improving the most, we would expect to see them improving the most in the current period too.) for these models:

Aspect	**QDiD**	**CiC**
Treatment of Time and Group	Symmetric	Asymmetric
QTET Computation	Not inherently scale-invariant	Outcome Variable Scale-Invariant

```
jt.qdid <- QDiD(
    re ~ treat,
    t = 1978,
    tmin1 = 1975,
    tname = "year",
    idname = "id",
    data = lalonde.psid.panel,
    iters = 10,
    panel = T
)

jt.cic <- CiC(
    re ~ treat,
    t = 1978,
    tmin1 = 1975,
```

```
    tname = "year",
    idname = "id",
    data = lalonde.psid.panel,
    iters = 10,
    panel = T
)
```

Chapter 31

Synthetic Control

Examples in marketing:

- (Tirunillai and Tellis, 2017): offline TV ad on Online Chatter
- (Wang et al., 2019): mobile hailing technology adoption on drivers' hourly earnings
- (Guo et al., 2020): payment disclosure laws effect on physician prescription behavior using Timing of the Massachusetts open payment law as the exogenous shock
- (Adalja et al., 2023): mandatory GMO labels had no impact on consumer demand (Using Vermont as a mandatory state)

Notes

- The SC method provides asymptotically normal estimators for various linear panel data models, given sufficiently large pre-treatment periods, making it a natural alternative to the Difference-in-differences model (Arkhangelsky and Hirshberg, 2023).
- SCM is **superior** than Matching Methods because it not only matches on covariates (i.e., pre-treatment variables), but also outcomes.
- For a review of the method, see (Abadie, 2021)
- SCMs can also be used under the Bayesian framework (Bayesian Synthetic Control) where we do not have to impose any restrictive priori (Kim et al., 2020)
- Different from Matching Methods because SCMs match on the pre-treatment outcomes in each period while Matching Methods match on the number of covariates.
- A data driven procedure to construct more comparable control groups (i.e., black box).

- To do causal inference with control and treatment group using Matching Methods, you typically have to have similar covariates in the control and the treated groups. However, if you don't methods like Propensity Scores and DID can perform rather poorly (i.e., large bias).

Advantages over Difference-in-differences

1. Maximization of the observable similarity between control and treatment (maybe also unobservables)
2. Can also be used in cases where no untreated case with similar on matching dimensions with treated cases
3. Objective selection of controls.

Advantages over linear regression

- Regression weights for the estimator will be outside of [0,1] (because regression allows extrapolation), and it will not be sparse (i.e., can be less than 0).
- No extrapolation under SCMs
- Explicitly state the fit (i.e., the weight)
- Can be estimated without the post-treatment outcomes for the control group (can't p-hack)

Advantages:

1. From the selection criteria, researchers can understand the relative importance of each candidate
2. Post-intervention outcomes are not used in synthetic. Hence, you can't retro-fit.
3. Observable similarity between control and treatment cases is maximized

Disadvantages:

1. It's hard to argue for the weights you use to create the "synthetic control"

SCM is recommended when

1. Social events to evaluate large-scale program or policy
2. Only one treated case with several control candidates.

Assumptions

- Donor subject is a good match for the synthetic control (i.e., gap between the dependent of the donor subject and that of the synthetic control should be 0 before treatment)
- Only the treated subject undergoes the treatment and not any of the subjects in the donor pool.
- No other changes to the subjects during the whole window.

- The counterfactual outcome of the treatment group can be imputed in a **linear combination** of control groups.

Identification: The exclusion restriction is met conditional on the pre-treatment outcomes.

`Synth` provides an algorithm that finds weighted combination of the comparison units where the weights are chosen such that it best resembles the values of predictors of the outcome variable for the affected units before the intervention

Setting (notation followed professor Alberto Abadie)

- $J+1$ units in periods $1, \dots, T$
- The first unit is the treated one during $T_0+1, \dots, T$
- J units are called a donor pool
- Y_{it}^I is the outcome for unit i if it's exposed to the treatment during $T_0+1, \dots T$
- Y_{it}^N is the outcome for unit i if it's not exposed to the treatment

We try to estimate the effect of treatment on the treated unit

$$\tau_{1t} = Y_{1t}^I - Y_{1t}^N$$

where we observe the first treated unit already $Y_{1t}^I = Y_{1t}$

To construct the synthetic control unit, we have to find appropriate weight for each donor in the donor pool by finding $\mathbf{W} = (w_2, \dots, w_{J=1})'$ where

- $w_j \geq 0$ for $j = 2, \dots, J+1$
- $w_2 + \dots + w_{J+1} = 1$

The "appropriate" vector $\mathbf{W}$ here is constrained to

$$\min ||\mathbf{X}_1 - \mathbf{X}_0 \mathbf{W}||$$

where

- $\mathbf{X}_1$ is the $k \times 1$ vector of pre-treatment characteristics for the treated unit
- $\mathbf{X}_0$ is the $k \times J$ matrix of pre-treatment characteristics for the untreated units

For simplicity, researchers usually use

$$\begin{aligned} &\min ||\mathbf{X}_1 - \mathbf{X}_0 \mathbf{W}|| \\ &= (\sum_{h=1}^{k} v_h (X_{h1} - w_2 X - h2 - \dots - w_{J+1} X_{hJ+1})^{1/2} \end{aligned}$$

where

- $v_1, \dots, v_k$ is a vector positive constants that represent the predictive power of the k predictors on Y_{1t}^N (i.e., the potential outcome of the treated without treatment) and it can be chosen either explicitly by the researcher or by data-driven methods

For penalized synthetic control (Abadie and L'hour, 2021), the minimization problem becomes

$$\min_{\mathbf{W}} ||\mathbf{X}_1 - \sum_{j=2}^{J+1} W_j \mathbf{X}_j||^2 + \lambda \sum_{j=2}^{J+1} W_j ||\mathbf{X}_1 - \mathbf{X}_j||^2$$

where

- $W_j \geq 0$ and $\sum_{j=2}^{J+1} W_j = 1$
- $\lambda > 0$ balances over-fitting of the treated and minimize the sum of pairwise distances
  - $\lambda \to 0$: pure synthetic control (i.e solution for the unpenalized estimator)
  - $\lambda \to \infty$: nearest neighbor matching

Advantages:

- For $\lambda > 0$, you have unique and sparse solution
- Reduces the interpolation bias when averaging dissimilar units
- Penalized SC never uses dissimilar units

Then the synthetic control estimator is

$$\hat{\tau}_{1t} = Y_{1t} - \sum_{j=2}^{J+1} w_j^* Y_{jt}$$

where Y_{jt} is the outcome for unit j at time t

Consideration

Under the factor model (Abadie et al., 2010)

$$Y_{it}^N = {}_t\mathbf{Z}_i + {}_{t\ i} + \epsilon_{it}$$

where

- Z_i = observables
- μ_i = unobservables

- ϵ_{it} = unit-level transitory shock (i.e., random noise)

with assumptions of $\mathbf{W}^*$ such that

$$\sum_{j=2}^{J+1} w_j^* \mathbf{Z}_j = \mathbf{Z}_1$$

$$\dots$$

$$\sum_{j=2}^{J+1} w_j^* Y_{j1} = Y_{11}$$

$$\sum_{j=2}^{J+1} w_j^* Y_{jT_0} = Y_{1T_0}$$

Basically, we assume that the synthetic control is a good counterfactual when the treated unit is not exposed to the treatment.

Then,

- the bias bound depends on close fit, which is controlled by the ratio between ϵ_{it} (transitory shock) and T_0 (the number of pre-treatment periods). In other words, you should have good fit for Y_{1t} for pre-treatment period (i.e., T_0 should be large while small variance in ϵ_{it})
- When you have poor fit, you have to use bias correction version of the synthetic control. See (Arkhangelsky et al., 2019, Abadie (2021), Ben-Michael et al. (2020))
- Overfitting can be the result of small T_0 (the number of pre-treatment periods), large J (the number of units in the donor pool), and large ϵ_{it} (noise)
  - Mitigation: put only similar units (to the treated one) in the donor pool

To make inference, we have to create a permutation distribution (by iteratively reassigning the treatment to the units in the donor pool and estimate the placebo effects in each iteration). We say there is an effect of the treatment when the magnitude of value of the treatment effect on the treated unit is extreme relative to the permutation distribution.

It's recommended to use one-sided inference. And the permutation distribution is superior to the p-values alone (because sampling-based inference is hard under SCMs either because of undefined sampling mechanism or the sample is the population).

For benchmark (permutation) distribution (e.g., uniform), see (Firpo and Possebom, 2018)

31.1 Applications

31.1.1 Example 1

by Danilo Freire

```
# install.packages("Synth")
# install.packages("gsynth")
library("Synth")
library("gsynth")
```

simulate data for 10 states and 30 years. State A receives the treatment `T = 20` after year 15.

```
set.seed(1)
year         <- rep(1:30, 10)
state        <- rep(LETTERS[1:10], each = 30)
X1           <- round(rnorm(300, mean = 2, sd = 1), 2)
X2           <- round(rbinom(300, 1, 0.5) + rnorm(300), 2)
Y            <- round(1 + 2 * X1 + rnorm(300), 2)
df           <- as.data.frame(cbind(Y, X1, X2, state, year))
df$Y         <- as.numeric(as.character(df$Y))
df$X1        <- as.numeric(as.character(df$X1))
df$X2        <- as.numeric(as.character(df$X2))
df$year      <- as.numeric(as.character(df$year))
df$state.num <- rep(1:10, each = 30)
df$state     <- as.character(df$state)
df$`T`       <- ifelse(df$state == "A" & df$year >= 15, 1, 0)
df$Y         <- ifelse(df$state == "A" & df$year >= 15,
                       df$Y + 20, df$Y)
```

```
str(df)
#> 'data.frame':    300 obs. of  7 variables:
#>  $ Y        : num  2.29 4.51 2.07 8.87 4.37 1.32 8 7.49 6.98 3.72 ...
#>  $ X1       : num  1.37 2.18 1.16 3.6 2.33 1.18 2.49 2.74 2.58 1.69 ...
#>  $ X2       : num  1.96 0.4 -0.75 -0.56 -0.45 1.06 0.51 -2.1 0 0.54 ...
#>  $ state    : chr  "A" "A" "A" "A" ...
#>  $ year     : num  1 2 3 4 5 6 7 8 9 10 ...
#>  $ state.num: int  1 1 1 1 1 1 1 1 1 1 ...
#>  $ T        : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
dataprep.out <-
    dataprep(
        df,
        predictors            = c("X1", "X2"),
        dependent             = "Y",
        unit.variable         = "state.num",
        time.variable         = "year",
```

```
        unit.names.variable   = "state",
        treatment.identifier  = 1,
        controls.identifier   = c(2:10),
        time.predictors.prior = c(1:14),
        time.optimize.ssr     = c(1:14),
        time.plot             = c(1:30)
    )


synth.out <- synth(dataprep.out)
#>
#> X1, X0, Z1, Z0 all come directly from dataprep object.
#>
#>
#> ****************
#>  searching for synthetic control unit
#>
#>
#> ****************
#> ****************
#> ****************
#>
#> MSPE (LOSS V): 9.831789
#>
#> solution.v:
#>  0.3888387 0.6111613
#>
#> solution.w:
#>  0.1115941 0.1832781 0.1027237 0.312091 0.06096758 0.03509706 0.05893735 0.05746256 0.07784853
```

```
print(synth.tables   <- synth.tab(
        dataprep.res = dataprep.out,
        synth.res    = synth.out)
      )
#> $tab.pred
#>    Treated Synthetic Sample Mean
#> X1   2.028     2.028       2.017
#> X2   0.513     0.513       0.394
#>
#> $tab.v
#>    v.weights
#> X1 0.389
#> X2 0.611
#>
#> $tab.w
```

```
#>    w.weights unit.names unit.numbers
#> 2      0.112          B            2
#> 3      0.183          C            3
#> 4      0.103          D            4
#> 5      0.312          E            5
#> 6      0.061          F            6
#> 7      0.035          G            7
#> 8      0.059          H            8
#> 9      0.057          I            9
#> 10     0.078          J           10
#>
#> $tab.loss
#>            Loss W   Loss V
#> [1,] 9.761708e-12 9.831789
```

```
path.plot(synth.res    = synth.out,
          dataprep.res = dataprep.out,
          Ylab         = c("Y"),
          Xlab         = c("Year"),
          Legend       = c("State A","Synthetic State A"),
          Legend.position = c("topleft")
)

abline(v   = 15,
       lty = 2)
```

Gaps plot:

```
gaps.plot(synth.res    = synth.out,
          dataprep.res = dataprep.out,
          Ylab         = c("Gap"),
          Xlab         = c("Year"),
          Ylim         = c(-30, 30),
          Main         = ""
)

abline(v   = 15,
       lty = 2)
```

Alternatively, `gsynth` provides options to estimate iterative fixed effects, and handle multiple treated units at tat time.

Here, we use two=way fixed effects and bootstrapped standard errors

```
gsynth.out <- gsynth(
  Y ~ `T` + X1 + X2,
  data = df,
  index = c("state", "year"),
  force = "two-way",
  CV = TRUE,
  r = c(0, 5),
  se = TRUE,
  inference = "parametric",
  nboots = 1000,
  parallel = F # TRUE
)
```

```
#> Cross-validating ...
#>  r = 0; sigma2 = 1.13533; IC = 0.95632; PC = 0.96713; MSPE = 1.65502
#>  r = 1; sigma2 = 0.96885; IC = 1.54420; PC = 4.30644; MSPE = 1.33375
#>  r = 2; sigma2 = 0.81855; IC = 2.08062; PC = 6.58556; MSPE = 1.27341*
#>  r = 3; sigma2 = 0.71670; IC = 2.61125; PC = 8.35187; MSPE = 1.79319
#>  r = 4; sigma2 = 0.62823; IC = 3.10156; PC = 9.59221; MSPE = 2.02301
#>  r = 5; sigma2 = 0.55497; IC = 3.55814; PC = 10.48406; MSPE = 2.79596
#>
#>  r* = 2
#>
#>
Simulating errors .............
Bootstrapping ...
#> ..........
```

```
plot(gsynth.out)
```

```
plot(gsynth.out, type = "counterfactual")
```

```
plot(gsynth.out, type = "counterfactual", raw = "all")
```

```
# shows estimations for the control cases
```

31.1.2 Example 2

by Leihua Ye

```r
library(Synth)
data("basque")
dim(basque) #774*17
#> [1] 774  17
head(basque)
#>   regionno     regionname year   gdpcap sec.agriculture sec.energy sec.industry
#> 1        1 Spain (Espana) 1955 2.354542              NA         NA           NA
#> 2        1 Spain (Espana) 1956 2.480149              NA         NA           NA
#> 3        1 Spain (Espana) 1957 2.603613              NA         NA           NA
#> 4        1 Spain (Espana) 1958 2.637104              NA         NA           NA
#> 5        1 Spain (Espana) 1959 2.669880              NA         NA           NA
#> 6        1 Spain (Espana) 1960 2.869966              NA         NA           NA
#>   sec.construction sec.services.venta sec.services.nonventa school.illit
#> 1               NA                 NA                    NA           NA
#> 2               NA                 NA                    NA           NA
#> 3               NA                 NA                    NA           NA
#> 4               NA                 NA                    NA           NA
#> 5               NA                 NA                    NA           NA
#> 6               NA                 NA                    NA           NA
#>   school.prim school.med school.high school.post.high popdens invest
#> 1          NA         NA          NA               NA      NA     NA
#> 2          NA         NA          NA               NA      NA     NA
#> 3          NA         NA          NA               NA      NA     NA
#> 4          NA         NA          NA               NA      NA     NA
#> 5          NA         NA          NA               NA      NA     NA
#> 6          NA         NA          NA               NA      NA     NA
```

transform data to be used in `synth()`

```r
dataprep.out <- dataprep(
    foo = basque,
    predictors = c(
        "school.illit",
        "school.prim",
        "school.med",
        "school.high",
        "school.post.high",
        "invest"
    ),
    predictors.op =  "mean",
    # the operator
    time.predictors.prior = 1964:1969,
    #the entire time frame from the #beginning to the end
    special.predictors = list(
        list("gdpcap", 1960:1969,  "mean"),
```

```
        list("sec.agriculture", seq(1961, 1969, 2), "mean"),
        list("sec.energy", seq(1961, 1969, 2), "mean"),
        list("sec.industry", seq(1961, 1969, 2), "mean"),
        list("sec.construction", seq(1961, 1969, 2), "mean"),
        list("sec.services.venta", seq(1961, 1969, 2), "mean"),
        list("sec.services.nonventa", seq(1961, 1969, 2), "mean"),
        list("popdens", 1969,  "mean")
    ),
    dependent =  "gdpcap",
    # dv
    unit.variable =  "regionno",
    #identifying unit numbers
    unit.names.variable =  "regionname",
    #identifying unit names
    time.variable =  "year",
    #time-periods
    treatment.identifier = 17,
    #the treated case
    controls.identifier = c(2:16, 18),
    #the control cases; all others #except number 17
    time.optimize.ssr = 1960:1969,
    #the time-period over which to optimize
    time.plot = 1955:1997
) #the entire time period before/after the treatment
```

where

- X_1 = the control case before the treatment
- X_0 = the control cases after the treatment
- Z_1: the treatment case before the treatment
- Z_0: the treatment case after the treatment

```
synth.out = synth(data.prep.obj = dataprep.out, method = "BFGS")
#>
#> X1, X0, Z1, Z0 all come directly from dataprep object.
#>
#>
#> ****************
#>  searching for synthetic control unit
#>
#>
#> ****************
#> ****************
#> ****************
#>
```

```
#> MSPE (LOSS V): 0.008864606
#>
#> solution.v:
#>  0.02773094 1.194e-07 1.60609e-05 0.0007163836 1.486e-07 0.002423908 0.0587055 0.26
#>
#> solution.w:
#>  2.53e-08 4.63e-08 6.44e-08 2.81e-08 3.37e-08 4.844e-07 4.2e-08 4.69e-08 0.8508145
```

Calculate the difference between the real basque region and the synthetic control

```
gaps = dataprep.out$Y1plot - (dataprep.out$Y0plot
                                    %*% synth.out$solution.w)
gaps[1:3,1]
#>       1955       1956       1957
#> 0.15023473 0.09168035 0.03716475
```

```
synth.tables = synth.tab(dataprep.res = dataprep.out,
                         synth.res = synth.out)
names(synth.tables)
#> [1] "tab.pred" "tab.v"    "tab.w"    "tab.loss"
synth.tables$tab.pred[1:13,]
#>                                        Treated Synthetic Sample Mean
#> school.illit                            39.888   256.337     170.786
#> school.prim                           1031.742  2730.104    1127.186
#> school.med                              90.359   223.340      76.260
#> school.high                             25.728    63.437      24.235
#> school.post.high                        13.480    36.153      13.478
#> invest                                  24.647    21.583      21.424
#> special.gdpcap.1960.1969                 5.285     5.271       3.581
#> special.sec.agriculture.1961.1969        6.844     6.179      21.353
#> special.sec.energy.1961.1969             4.106     2.760       5.310
#> special.sec.industry.1961.1969          45.082    37.636      22.425
#> special.sec.construction.1961.1969       6.150     6.952       7.276
#> special.sec.services.venta.1961.1969    33.754    41.104      36.528
#> special.sec.services.nonventa.1961.1969  4.072     5.371       7.111
```

Relative importance of each unit

```
synth.tables$tab.w[8:14, ]
#>    w.weights            unit.names unit.numbers
#> 9      0.000    Castilla-La Mancha            9
#> 10     0.851               Cataluna           10
#> 11     0.000  Comunidad Valenciana           11
#> 12     0.000           Extremadura           12
#> 13     0.000               Galicia           13
#> 14     0.149 Madrid (Comunidad De)           14
#> 15     0.000    Murcia (Region de)           15
```

```
# plot the changes before and after the treatment
path.plot(
    synth.res = synth.out,
    dataprep.res = dataprep.out,
    Ylab = "real per-capita gdp (1986 USD, thousand)",
    Xlab = "year",
    Ylim = c(0, 12),
    Legend = c("Basque country",
               "synthetic Basque country"),
    Legend.position = "bottomright"
)
```

```
gaps.plot(
    synth.res = synth.out,
    dataprep.res = dataprep.out,
    Ylab =  "gap in real per - capita GDP (1986 USD, thousand)",
    Xlab =  "year",
    Ylim = c(-1.5, 1.5),
    Main = NA
)
```

Doubly Robust Difference-in-Differences

Example from `DRDID` package

```
library(DRDID)
data(nsw_long)
# Form the Lalonde sample with CPS comparison group
eval_lalonde_cps <- subset(nsw_long, nsw_long$treated == 0 |
                                nsw_long$sample == 2)
```

Estimate Average Treatment Effect on Treated using Improved Locally Efficient Doubly Robust DID estimator

```
out <-
    drdid(
        yname = "re",
        tname = "year",
        idname = "id",
        dname = "experimental",
        xformla = ~ age + educ + black + married + nodegree + hisp + re74,
        data = eval_lalonde_cps,
        panel = TRUE
    )
summary(out)
#>  Call:
#> drdid(yname = "re", tname = "year", idname = "id", dname = "experimental",
#>     xformla = ~age + educ + black + married + nodegree + hisp +
#>         re74, data = eval_lalonde_cps, panel = TRUE)
#> ------------------------------------------------------------------
```

```
#>  Further improved locally efficient DR DID estimator for the ATT:
#>
#>    ATT     Std. Error  t value    Pr(>|t|)  [95% Conf. Interval]
#> -901.2703   393.6247   -2.2897     0.022    -1672.7747  -129.766
#> ------------------------------------------------------------------
#>  Estimator based on panel data.
#>  Outcome regression est. method: weighted least squares.
#>  Propensity score est. method: inverse prob. tilting.
#>  Analytical standard error.
#> -------------------------------------------------------------------
#>  See Sant'Anna and Zhao (2020) for details.
```

31.1.3 Example 3

by `Synth` package's authors

```r
library(Synth)
data("basque")
```

`synth()` requires

- X_1 vector of treatment predictors
- X_0 matrix of same variables for control group
- Z_1 vector of outcome variable for treatment group
- Z_0 matrix of outcome variable for control group

use `dataprep()` to prepare data in the format that can be used throughout the `Synth` package

```r
dataprep.out <- dataprep(
    foo = basque,
    predictors = c(
        "school.illit",
        "school.prim",
        "school.med",
        "school.high",
        "school.post.high",
        "invest"
    ),
    predictors.op = "mean",
    time.predictors.prior = 1964:1969,
    special.predictors = list(
        list("gdpcap", 1960:1969 , "mean"),
        list("sec.agriculture", seq(1961, 1969, 2), "mean"),
        list("sec.energy", seq(1961, 1969, 2), "mean"),
        list("sec.industry", seq(1961, 1969, 2), "mean"),
```

```
        list("sec.construction", seq(1961, 1969, 2), "mean"),
        list("sec.services.venta", seq(1961, 1969, 2), "mean"),
        list("sec.services.nonventa", seq(1961, 1969, 2), "mean"),
        list("popdens", 1969, "mean")
    ),
    dependent = "gdpcap",
    unit.variable = "regionno",
    unit.names.variable = "regionname",
    time.variable = "year",
    treatment.identifier = 17,
    controls.identifier = c(2:16, 18),
    time.optimize.ssr = 1960:1969,
    time.plot = 1955:1997
)
```

find optimal weights that identifies the synthetic control for the treatment group

```
synth.out <- synth(data.prep.obj = dataprep.out, method = "BFGS")
#>
#> X1, X0, Z1, Z0 all come directly from dataprep object.
#>
#>
#> ****************
#>  searching for synthetic control unit
#>
#>
#> ****************
#> ****************
#> ****************
#>
#> MSPE (LOSS V): 0.008864606
#>
#> solution.v:
#>  0.02773094 1.194e-07 1.60609e-05 0.0007163836 1.486e-07 0.002423908 0.0587055 0.26
#>
#> solution.w:
#>  2.53e-08 4.63e-08 6.44e-08 2.81e-08 3.37e-08 4.844e-07 4.2e-08 4.69e-08 0.8508145
```

```
gaps <- dataprep.out$Y1plot - (dataprep.out$Y0plot %*% synth.out$solution.w)
gaps[1:3, 1]
#>       1955       1956       1957
#> 0.15023473 0.09168035 0.03716475
```

```
synth.tables <-
    synth.tab(dataprep.res = dataprep.out, synth.res = synth.out)
names(synth.tables) # you can pick tables to see
```

```
#> [1] "tab.pred" "tab.v"    "tab.w"    "tab.loss"
```

```
path.plot(
    synth.res = synth.out,
    dataprep.res = dataprep.out,
    Ylab = "real per-capita GDP (1986 USD, thousand)",
    Xlab = "year",
    Ylim = c(0, 12),
    Legend = c("Basque country",
               "synthetic Basque country"),
    Legend.position = "bottomright"
)
```

```
gaps.plot(
    synth.res = synth.out,
    dataprep.res = dataprep.out,
    Ylab = "gap in real per-capita GDP (1986 USD, thousand)",
    Xlab = "year",
    Ylim = c(-1.5, 1.5),
    Main = NA
)
```

You could also run placebo tests

31.1.4 Example 4

by Michael Robbins and Steven Davenport who are authors of `MicroSynth` with the following improvements:

- Standardization `use.survey = TRUE` and permutation (`perm = 250` and `jack = TRUE`) for placebo tests
- Omnibus statistic (set to `omnibus.var`) for multiple outcome variables
- incorporate multiple follow-up periods `end.post`

Notes:

- Both predictors and outcome will be used to match units before intervention
  - Outcome variable has to be **time-variant**
  - Predictors are **time-invariant**

```
# right now the package is not availabe for R version 4.2
library(microsynth)
data("seattledmi")


cov.var <-
    c(
        "TotalPop",
```

```
        "BLACK",
        "HISPANIC",
        "Males_1521",
        "HOUSEHOLDS",
        "FAMILYHOUS",
        "FEMALE_HOU",
        "RENTER_HOU",
        "VACANT_HOU"
    )
match.out <- c("i_felony", "i_misdemea", "i_drugs", "any_crime")
```

```
sea1 <- microsynth(
    seattledmi,
    idvar       = "ID",
    timevar     = "time",
    intvar      = "Intervention",
    start.pre   = 1,
    end.pre     = 12,
    end.post    = 16,
    match.out   = match.out, # outcome variable will be matched on exactly
    match.covar = cov.var, # specify covariates will be matched on exactly
    result.var  = match.out, # used to report results
    omnibus.var = match.out, # feature in the omnibus p-value
    test        = "lower",
    n.cores     = min(parallel::detectCores(), 2)
)
sea1
summary(sea1)
```

```
plot_microsynth(sea1)
```

```
sea2 <- microsynth(
    seattledmi,
    idvar = "ID",
    timevar = "time",
    intvar = "Intervention",
    start.pre = 1,
    end.pre = 12,
    end.post = c(14, 16),
    match.out = match.out,
    match.covar = cov.var,
    result.var = match.out,
    omnibus.var = match.out,
    test = "lower",
    perm = 250,
    jack = TRUE,
```

```
    n.cores = min(parallel::detectCores(), 2)
)
```

31.2 Augmented Synthetic Control Method

package: `augsynth` (Ben-Michael et al., 2021)

31.3 Synthetic Control with Staggered Adoption

references: https://ebenmichael.github.io/assets/research/jamboree.pdf (Ben-Michael et al., 2022) package: `augsynth`

31.4 Bayesian Synthetic Control

Kim et al. (2020)

Pang et al. (2022)

31.5 Generalized Synthetic Control

reference: (Xu, 2017)

- Bootstrap procedure here is biased (Li and Sonnier, 2023). Hence, we need to follow Li and Sonnier (2023) in terms of SEs estimation.

31.6 Other Advances

Sun et al. (2023) Using Multiple Outcomes to Improve SCM

- **Common Weights Across Outcomes**: This paper proposes using a single set of synthetic control weights across multiple outcomes, rather than estimating separate weights for each outcome.
- **Reduced Bias with Low-Rank Factor Model**: By balancing a vector or an index of outcomes, this approach yields lower bias bounds under a low-rank factor model, with further improvements as the number of outcomes increases.
- **Evidence**: re-analysis of the Flint water crisis's impact on educational outcome.

Chapter 32

Event Studies

The earliest paper that used event study was (Dolley, 1933)

(Campbell et al., 1998) introduced this method, which based on the efficient markets theory by (Fama, 1970)

Review:

- (McWilliams and Siegel, 1997): in management
- (Sorescu et al., 2017): in marketing

Previous marketing studies:

1. Firm-initiated activities

- (Horsky and Swyngedouw, 1987): name change
- (Chaney et al., 1991) new product announcements
- (Agrawal and Kamakura, 1995): celebrity endorsement
- (Lane and Jacobson, 1995): brand extensions
- (Houston and Johnson, 2000): joint venture
- (Geyskens et al., 2002): Internet channel (for newspapers)
- (Cornwell et al., 2005): sponsorship announcements
- (Elberse, 2007): casting announcements
- (Sorescu et al., 2007): M&A
- (Sood and Tellis, 2009): innovation payoff
- (Wiles and Danielova, 2009): product placements in movies
- (Joshi and Hanssens, 2009): movie releases

- (Wiles et al., 2010): Regulatory Reports of Deceptive Advertising
- (Boyd et al., 2010): new CMO appointments
- (Karniouchina et al., 2011): product placement
- (Wiles et al., 2012): Brand Acquisition and Disposal
- (Kalaignanam and Bahadir, 2013): corporate brand name change
- (Raassens et al., 2012): new product development outsourcing
- (Mazodier and Rezaee, 2013): sports announcements
- (Borah and Tellis, 2014): make, buy or ally for innovations
- (Homburg et al., 2014): channel expansions
- (Fang et al., 2015): Co-development agreements
- (Wu et al., 2015): horizontal collaboration in new product development
- (Fama et al., 1969): stock split

2. Non-firm-initiated activities

- (Sorescu et al., 2003): FDA approvals
- (Pandey et al., 2005): diversity elite list
- (Balasubramanian et al., 2005): high-quality achievements
- (Tellis and Johnson, 2007): quality reviews by Walter Mossberg
- (Fornell et al., 2006): customer satisfaction
- (Gielens et al., 2008): Walmart's entry into the UK market
- (Boyd and Spekman, 2008): indirect ties
- (Rao et al., 2008): FDA approvals
- (Ittner et al., 2009): customer satisfaction
- (Tipton et al., 2009): Deceptive advertising
- (Chen et al., 2009b): product recalls
- (Jacobson and Mizik, 2009): satisfaction score release
- (Karniouchina et al., 2009): Mad money with Jim Cramer
- (Wiles et al., 2010): deceptive advertising
- (Chen et al., 2012): third-party movie reviews
- (Xiong and Bharadwaj, 2013): positive and negative news
- (Gao et al., 2015): product recall
- (Malhotra and Kubowicz Malhotra, 2011): data breach

- (Bhagat et al., 1998): litigation

Potential avenues:

- Ad campaigns
- Market entry
- product failure/recalls
- Patents

Pros:

- Better than accounting based measures (e.g., profits) because managers can manipulate profits (Benston, 1985)
- Easy to do

Fun fact:

- (Dubow and Monteiro, 2006) came up with a way to gauge how 'clean' a market is. They based their measure on how much prices seemed to move in a way that suggested insider knowledge, before the release of important regulatory announcements that could affect the stock prices. Such price shifts might suggest that insider trading was occurring. Essentially, they were watching for any unusual price changes before the day of the announcement.

Events can be

- Internal (e.g., stock repurchase)
- External (e.g., macroeconomic variables)

Assumptions:

1. Efficient market theory
2. Shareholders are the most important group among stakeholders
3. The event sharply affects share price
4. Expected return is calculated appropriately

Steps:

1. Event Identification: (e.g., dividends, M&A, stock buyback, laws or regulation, privatization vs. nationalization, celebrity endorsements, name changes, or brand extensions etc. To see the list of events in US and international, see WRDS **S&P Capital IQ Key Developments**). Events must affect either cash flows or on the discount rate of firms (Sorescu et al., 2017, p. 191)
    1. Estimation window: Normal return expected return ($T_0 \to T_1$) (sometimes include days before to capture leakages).

- Recommendation by (Johnston, 2007) is to use 250 days before the event (and 45-day between the estimation window and the event window).
  - (Wiles et al., 2012) used an 90-trading-day estimation window ending 6 days before the event (this is consistent with the finance literature).
  - (Gielens et al., 2008) 260 to 10 days before or 300 to 46 days before
  - (Tirunillai and Tellis, 2012) estimation window of 255 days and ends 46 days before the event.
- Similarly, (McWilliams and Siegel, 1997) and (Fornell et al., 2006) 255 days ending 46 days before the event date
- (Sorescu et al., 2017, p. 194) suggest 100 days before the event date
- Leakage: try to cover as broad news sources as possible (LexisNexis, Factiva, and RavenPack).

2. Event window: contain the event date ($T_1 \to T_2$) (have to argue for the event window and can't do it empirically)
   - One day: (Balasubramanian et al., 2005; Boyd et al., 2010; Fornell et al., 2006)
   - Two days: (Raassens et al., 2012; Sood and Tellis, 2009)
   - Up to 10 days: (Cornwell et al., 2005; Kalaignanam and Bahadir, 2013; Sorescu et al., 2007)
3. Post Event window: $T_2 \to T_3$

2. Normal vs. Abnormal returns

$$\epsilon_{it}^* = \frac{P_{it} - E(P_{it})}{P_{it-1}} = R_{it} - E(R_{it}|X_t)$$

where

- ϵ_{it}^* = abnormal return
- R_{it} = realized (actual) return
- P = dividend-adjusted price of the stock
- $E(R_{it}|X_t)$ normal expected return

There are several model to calculate the expected return

A. Statistical Models: assumes jointly multivariate normal and iid over time (need distributional assumptions for valid finite-sample estimation) rather robust (hence, recommended)

1. Constant Mean Return Model
2. Market Model
3. Adjusted Market Return Model
4. Factor Model

B. Economic Model (strong assumption regarding investor behavior)

1. Capital Asset Pricing Model (CAPM)
2. Arbitrage Pricing Theory (APT)

32.1 Other Issues

32.1.1 Event Studies in marketing

(Skiera et al., 2017) What should be the dependent variable in marketing-related event studies?

- Based on valuation theory, Shareholder value = the value of the operating business + non-operating asset - debt (Schulze et al., 2012)
  - Many marketing events only affect the operating business value, but not non-operating assets and debt
- Ignoring the differences in firm-specific leverage effects has dual effects:
  - inflates the impact of observation pertaining to firms with large debt
  - deflates those pertaining to firms with large non-operating asset.
- It's recommended that marketing papers should report both CAR^{OB} and CAR^{SHV} and argue for whichever one more appropriate.
- Up until this paper, only two previous event studies control for financial structure: (Gielens et al., 2008) (Chaney et al., 1991)

Definitions:

- Cumulative abnormal percentage return on shareholder value (CAR^{SHV})
  - Shareholder value refers to a firm's market capitalization = share price x # of shares.
- Cumulative abnormal percentage return on the value of the operating business (CAR^{OB})
  - $CAR^{OB} = CAR^{SHV}$/leverage effect$_{before}$
  - Leverage effect = Operating business value / Shareholder value (LE describes how a 1% change in operating business translates into a percentage change in shareholder value).
  - Value of operating business = shareholder value - non-operating assets + debt
  - Leverage effect $\neq$ leverage ratio, where leverage ratio is debt / firm size
    - debt = long-term + short-term debt; long-term debt

    * firm size = book value of equity; market cap; total assets; debt + equity

- Operating assets are those used by firm in their core business operations (e..g, property, plant, equipment, natural resources, intangible asset)
- Non–operating assets (redundant assets), do not play a role in a firm's operations, but still generate some form of return (e.g., excess cash , marketable securities - commercial papers, market instruments)

Marketing events usually influence the value of a firm's operating assets (more specifically intangible assets). Then, changes in the value of the operating business can impact shareholder value.

- Three rare instances where marketing events can affect non-operating assets and debt
  - (Hall et al., 2004): excess pre-orderings can influence short-term debt
  - (Berger et al., 1997) Firing CMO increase debt as the manager's tenure is negatively associated with the firm's debt
  - (Bhaduri, 2002) production of unique products.

A marketing-related event can either influence

- value components of a firm's value (= firm's operating business, non-operating assets and its debt)
- only the operating business.

Replication of the leverage effect

$$\begin{aligned}\text{leverage effect} &= \frac{\text{operating business}}{\text{shareholder value}} \\ &= \frac{\text{(shareholder value - non-operating assets + debt)}}{\text{shareholder value}} \\ &= \frac{prcc_f \times csho - ivst + dd1 + dltt + pstk}{prcc_f \times csho}\end{aligned}$$

Compustat Data Item

Label	Variable
`prcc_f`	Share price
`csho`	Common shares outstanding
`ivst`	short-term investments (Non-operating assets)
`dd1`	long-term debt due in one year
`dltt`	long-term debt
`pstk`	preferred stock

Since WRDS no longer maintains the S&P 500 list as of the time of this writing, I can't replicate the list used in (Skiera et al., 2017) paper.

```
library(tidyverse)
df_leverage_effect <- read.csv("data/leverage_effect.csv.gz") %>%

    # get active firms only
    filter(costat == "A") %>%

    # drop missing values
    drop_na() %>%

    # create the leverage effect variable
    mutate(le = (prcc_f * csho - ivst + dd1 + dltt + pstk)/ (prcc_f * csho)) %>%

    # get shareholder value
    mutate(shv = prcc_f * csho) %>%

    # remove Infinity value for leverage effect (i.e., shareholder value = 0)
    filter_all(all_vars(!is.infinite(.))) %>%

    # positive values only
    filter_all(all_vars(. > 0)) %>%

    # get the within coefficient of variation
    group_by(gvkey) %>%
    mutate(within_var_mean_le = mean(le),
           within_var_sd_le = sd(le)) %>%
    ungroup()


# get the mean and standard deviation
mean(df_leverage_effect$le)
#> [1] 150.1087
max(df_leverage_effect$le)
#> [1] 183629.6
hist(df_leverage_effect$le)
```

```
# coefficient of variation
sd(df_leverage_effect$le) / mean(df_leverage_effect$le) * 100
#> [1] 2749.084

# Within-firm variation (similar to fig 3a)
df_leverage_effect %>%
    group_by(gvkey) %>%
    slice(1) %>%
    ungroup() %>%
    dplyr::select(within_var_mean_le, within_var_sd_le) %>%
    dplyr::mutate(cv = within_var_sd_le/ within_var_mean_le) %>%
    dplyr::select(cv) %>%
    pull() %>%
    hist()
```

32.1.2 Economic significance

Total wealth gain (loss) from the event

$$\Delta W_t = CAR_t \times MKTVAL_0$$

where

- ΔW_t = gain (loss)
- CAR_t = cumulative residuals to date t
- $MKTVAL_0$ market value of the firm before the event window

32.1.3 Statistical Power

increases with

- more firms
- less days in the event window (avoiding potential contamination from confounds)

32.2 Testing

32.2.1 Parametric Test

(Brown and Warner, 1985) provide evidence that even in the presence of non-normality, the parametric tests still perform well. Since the proportion of posi-

tive and negative abnormal returns tends to be equal in the sample (of at least 5 securities). The excess returns will coverage to normality as the sample size increases. Hence, parametric test is advocated than non-parametric one.

Low power to detect significance (Kothari and Warner, 1997)

- Power = f(sample, size, the actual size of abnormal returns, the variance of abnormal returns across firms)

32.2.1.1 T-test

Applying CLT

$$t_{CAR} = \frac{C\bar{A}R_{it}}{\sigma(CAR_{it})/\sqrt{n}}$$
$$t_{BHAR} = \frac{BH\bar{A}R_{it}}{\sigma(BHAR_{it})/\sqrt{n}}$$

Assume

- Abnormal returns are normally distributed
- Var(abnormal returns) are equal across firms
- No cross-correlation in abnormal returns.

Hence, it will be misspecified if you suspected

- Heteroskedasticity
- Cross-sectional dependence
- Technically, abnormal returns could follow non-normal distribution (but because of the design of abnormal returns calculation, it typically forces the distribution to be normal)

To address these concerns, Patell Standardized Residual (PSR) can sometimes help.

32.2.1.2 Patell Standardized Residual (PSR)

(Patell, 1976)

- Since market model uses observations outside the event window, abnormal returns contain prediction errors on top of the true residuals , and should be standardized:

$$AR_{it} = \frac{\hat{u}_{it}}{s_i\sqrt{C_{it}}}$$

where

- $\hat{u}_{it}$ = estimated residual
- s_i = standard deviation estimate of residuals (from the estimation period)
- C_{it} = a correction to account for the prediction's increased variation outside of the estimation period (Strong, 1992)

$$C_{it} = 1 + \frac{1}{T} + \frac{(R_{mt} - \bar{R}_m)^2}{\sum_t (R_{mt} - \bar{R}_m)^2}$$

where

- T = number of observations (from estimation period)
- R_{mt} = average rate of return of all stocks trading the the stock market at time t
- $\bar{R}_m = \frac{1}{T} \sum_{t=1}^{T} R_{mt}$

32.2.2 Non-parametric Test

- No assumptions about return distribution
- Sign Test (assumes symmetry in returns)
  - `binom.test()`
- Wilcoxon Signed-Rank Test (allows for non-symmetry in returns)
  - Use `wilcox.test(sample)`
- Gen Sign Test
- Corrado Rank Test

32.3 Sample

- Sample can be relative small
  - (Wiles et al., 2012) 572 acquisition announcements, 308 disposal announcements
  - Can range from 71 (Markovitch and Golder, 2008) to 3552 (Borah and Tellis, 2014)

32.3.1 Confounders

- Avoid confounding events: earnings announcements, key executive changes, unexpected stock buybacks, changes in dividends within the two-trading day window surrounding the event, mergers and acquisitions, spin-offers, stock splits, management changes, joint ventures, unexpected

dividend, IPO, debt defaults, dividend cancellations (McWilliams and Siegel, 1997)

According to (Fornell et al., 2006), need to control:

- one-day event period = day when Wall Street Journal publish ACSI announcement.
- 5 days before and after event to rule out other news (PR Newswires, Dow Jones, Business Wires)
  - M&A, Spin-offs, stock splits
  - CEO or CFO changes,
  - Layoffs, restructurings, earnings announcements, lawsuits
  - Capital IQ - Key Developments: covers almost all important events so you don't have to search on news.

(Sorescu et al., 2017) examine confounding events in the short-term windows:

- From RavenPack, 3982 US publicly traded firms, with all the press releases (2000-2013)
- 3-day window around event dates
- The difference between a sample with full observations and a sample without confounded events is negligible (non-significant).
- Conclusion: **excluding confounded observations may be unnecessary for short-term event studies.**
  - Biases can stem from researchers pick and choose events to exclude
  - As time progresses, more and more events you need to exclude which can be infeasible.

To further illustrate this point, let's do a quick simulation exercise

In this example, we will explore three types of events:

1. Focal events
2. Correlated events (i.e., events correlated with the focal events; the presence of correlated events can follow the presence of the focal event)
3. Uncorrelated events (i.e., events with dates that might randomly coincide with the focal events, but are not correlated with them).

We have the ability to control the strength of correlation between focal and correlated events in this study, as well as the number of unrelated events we wish to examine.

Let's examine the implications of including and excluding correlated and uncorrelated events on the estimates of our focal events.

```r
# Load required libraries
library(dplyr)
library(ggplot2)
library(tidyr)
library(tidyverse)

# Parameters
n                  <- 100000         # Number of observations
n_focal            <- round(n * 0.2) # Number of focal events
overlap_correlated <- 0.5            # Overlapping percentage between focal and correlated events

# Function to compute mean and confidence interval
mean_ci <- function(x) {
    m <- mean(x)
    ci <- qt(0.975, length(x)-1) * sd(x) / sqrt(length(x)) # 95% confidence interval
    list(mean = m, lower = m - ci, upper = m + ci)
}

# Simulate data
set.seed(42)
data <- tibble(
    date       = seq.Date(from = as.Date("2010-01-01"), by = "day", length.out = n), # Date seque
    focal      = rep(0, n),
    correlated = rep(0, n),
    ab_ret     = rnorm(n)
)


# Define focal events
focal_idx <- sample(1:n, n_focal)
data$focal[focal_idx] <- 1

true_effect <- 0.25

# Adjust the ab_ret for the focal events to have a mean of true_effect
data$ab_ret[focal_idx] <- data$ab_ret[focal_idx] - mean(data$ab_ret[focal_idx]) + true_effect



# Determine the number of correlated events that overlap with focal and those that don't
n_correlated_overlap <- round(length(focal_idx) * overlap_correlated)
n_correlated_non_overlap <- n_correlated_overlap

# Sample the overlapping correlated events from the focal indices
correlated_idx <- sample(focal_idx, size = n_correlated_overlap)
```

```
# Get the remaining indices that are not part of focal
remaining_idx <- setdiff(1:n, focal_idx)

# Check to ensure that we're not attempting to sample more than the available remainin
if (length(remaining_idx) < n_correlated_non_overlap) {
    stop("Not enough remaining indices for non-overlapping correlated events")
}

# Sample the non-overlapping correlated events from the remaining indices
correlated_non_focal_idx <- sample(remaining_idx, size = n_correlated_non_overlap)

# Combine the two to get all correlated indices
all_correlated_idx <- c(correlated_idx, correlated_non_focal_idx)

# Set the correlated events in the data
data$correlated[all_correlated_idx] <- 1


# Inflate the effect for correlated events to have a mean of
correlated_non_focal_idx <- setdiff(all_correlated_idx, focal_idx) # Fixing the select
data$ab_ret[correlated_non_focal_idx] <- data$ab_ret[correlated_non_focal_idx] - mean(


# Define the numbers of uncorrelated events for each scenario
num_uncorrelated <- c(5, 10, 20, 30, 40)

# Define uncorrelated events
for (num in num_uncorrelated) {
    for (i in 1:num) {
        data[paste0("uncorrelated_", i)] <- 0
        uncorrelated_idx <- sample(1:n, round(n * 0.1))
        data[uncorrelated_idx, paste0("uncorrelated_", i)] <- 1
    }
}


# Define uncorrelated columns and scenarios
unc_cols <- paste0("uncorrelated_", 1:num_uncorrelated)
results <- tibble(
    Scenario = c("Include Correlated", "Correlated Effects", "Exclude Correlated", "Exc
    MeanEffect = c(
        mean_ci(data$ab_ret[data$focal == 1])$mean,
        mean_ci(data$ab_ret[data$focal == 0 | data$correlated == 1])$mean,
        mean_ci(data$ab_ret[data$focal == 1 & data$correlated == 0])$mean,
        mean_ci(data$ab_ret[data$focal == 1 & data$correlated == 0 & rowSums(data[, pas
```

```
    ),
    LowerCI = c(
        mean_ci(data$ab_ret[data$focal == 1])$lower,
        mean_ci(data$ab_ret[data$focal == 0 | data$correlated == 1])$lower,
        mean_ci(data$ab_ret[data$focal == 1 & data$correlated == 0])$lower,
        mean_ci(data$ab_ret[data$focal == 1 & data$correlated == 0 & rowSums(data[, paste0("uncor
    ),
    UpperCI = c(
        mean_ci(data$ab_ret[data$focal == 1])$upper,
        mean_ci(data$ab_ret[data$focal == 0 | data$correlated == 1])$upper,
        mean_ci(data$ab_ret[data$focal == 1 & data$correlated == 0])$upper,
        mean_ci(data$ab_ret[data$focal == 1 & data$correlated == 0 & rowSums(data[, paste0("uncor
    )
)

# Add the scenarios for excluding 5, 10, 20, and 50 uncorrelated
for (num in num_uncorrelated) {
    unc_cols <- paste0("uncorrelated_", 1:num)
    results <- results %>%
        add_row(
            Scenario = paste("Exclude", num, "Uncorrelated"),
            MeanEffect = mean_ci(data$ab_ret[data$focal == 1 & data$correlated == 0 & rowSums(dat
            LowerCI = mean_ci(data$ab_ret[data$focal == 1 & data$correlated == 0 & rowSums(data[,
            UpperCI = mean_ci(data$ab_ret[data$focal == 1 & data$correlated == 0 & rowSums(data[,
        )
}


ggplot(results,
       aes(
           x = factor(Scenario, levels = Scenario),
           y = MeanEffect,
           ymin = LowerCI,
           ymax = UpperCI
       )) +
    geom_pointrange() +
    coord_flip() +
    ylab("Mean Effect") +
    xlab("Scenario") +
    ggtitle("Mean Effect of Focal Events under Different Scenarios") +
    geom_hline(yintercept = true_effect,
               linetype = "dashed",
               color = "red")
```

As depicted in the plot, the inclusion of correlated events demonstrates minimal impact on the estimation of our focal events. Conversely, excluding these correlated events can diminish our statistical power. This is true in cases of pronounced correlation.

However, the consequences of excluding unrelated events are notably more significant. It becomes evident that by omitting around 40 unrelated events from our study, we lose the ability to accurately identify the true effects of the focal events. In reality and within research, we often rely on the Key Developments database, excluding over 150 events, a practice that can substantially impair our capacity to ascertain the authentic impact of the focal events.

This little experiment really drives home the point – you better have a darn good reason to exclude an event from your study (make it super convincing)!

32.4 Biases

- Different closing time obscure estimation of the abnormal returns, check (Campbell et al., 1998)
- Upward bias in aggregating CAR + transaction prices (bid and ask)
- Cross-sectional dependence in the returns bias the standard deviation estimates downward, which inflates the test statistics when events share common dates (MacKinlay, 1997). Hence, (Jaffe, 1974) Calendar-time Portfolio Abnormal Returns (CTARs) should be used to correct for this bias.
- (Wiles et al., 2012): For events confined to relatively few industries, cross-

sectional dependence in the returns can bias the SD estimate downward, inflating the associated test statistics" (p. 47). To control for potential cross-sectional correlation in the abnormal returns, you can use time-series standard deviation test statistic (Brown and Warner, 1980)

- Sample selection bias (self-selection of firms into the event treatment) similar to omitted variable bias where the omitted variable is the private info that leads a firm to take the action.
  - See Endogenous Sample Selection for more methods to correct this bias.
  - Use Heckman model (Acharya, 1993)
    - But hard to find an instrument that meets the exclusion requirements (and strong, because weak instruments can lead to multicollinearity in the second equation)
    - Can estimate the private information unknown to investors (which is Mills ratio λ itself). Testing λ significance is to see whether private info can explain outcomes (e.g., magnitude of the CARs to the announcement).
    - Examples: (Chen et al., 2009b) (Wiles et al., 2012) (Fang et al., 2015)
  - Counterfactual observations
    - Propensity score matching:
      - Finance: (Iskandar-Datta and Jia, 2013, Doan and Iskandar-Datta (2021)) (Masulis and Nahata, 2011)
      - Marketing: (Warren and Sorescu, 2017) (Borah and Tellis, 2014) (Cao and Sorescu, 2013)
    - Switching regression: comparison between 2 specific outcomes (also account for selection on unobservables - using instruments) (Cao and Sorescu, 2013)

32.5 Long-run event studies

- Usually make an assumption that the distribution of the abnormal returns to these events has a mean of 0 (Sorescu et al., 2017, p. 192). And (Sorescu et al., 2017) provide evidence that for all events they examine the results from samples with and without confounding events do not differ.
- Long-horizon event studies face challenges due to systematic errors over time and sensitivity to model choice.

- Two main approaches are used to measure long-term abnormal stock returns
  - Buy and Hold Abnormal Returns (BHAR)
  - Long-term Cumulative Abnormal Returns (LCARs)
  - Calendar-time Portfolio Abnormal Returns (CTARs) (Jensen's Alpha): manages cross-sectional dependence better and is less sensitive to (asset pricing) model misspecification
- Two types:
  - Unexpected changes in firm specific variables (typically not announced, may not be immediately visible to all investors, impact on firm value is not straightforward): customer satisfaction scores effect on firm value (Jacobson and Mizik, 2009) or unexpected changes in marketing expenditures (Kim and McAlister, 2011) to determine mispricing.
  - Complex consequences (investors take time to learn and incorporate info): acquisition depends on integration (Sorescu et al., 2007)
- 12 - 60 months event window: (Loughran and Ritter, 1995) (Brav and Gompers, 1997)
- Example: (Dutta et al., 2018)

```r
library(crseEventStudy)

# example by the package's author
data(demo_returns)
SAR <-
    sar(event = demo_returns$EON,
        control = demo_returns$RWE,
        logret = FALSE)
mean(SAR)
#> [1] 0.006870196
```

32.5.1 Buy and Hold Abnormal Returns (BHAR)

- Classic references: (Loughran and Ritter, 1995) (Barber and Lyon, 1997) (Lyon et al., 1999)

Use a portfolio of stocks that are close matches of the current firm over the same period as benchmark, and see the difference between the firm return and that of the portfolio.

- More technical note is that it measures returns from buying stocks in event-experiencing firms and shorting stocks in similar non-event firms within the same time.

- Because of high cross-sectional correlations, BHARs' t-stat can be inflated, but its rank order is not affected (Markovitch and Golder, 2008; Sorescu et al., 2007)

To construct the portfolio, use similar

- size
- book-to-market
- momentum

Matching Procedure (Barber and Lyon, 1997):

1. Each year from July to June, all common stocks in the CRSP database are categorized into ten groups (deciles) based on their market capitalization from the previous June.

2. Within these deciles, firms are further sorted into five groups (quintiles) based on their book-to-market ratios as of December of the previous year or earlier, considering possible delays in financial statement reporting.

3. Benchmark portfolios are designed to exclude firms with specific events but include all firms that can be classified into the characteristic-based portfolios.

Similarly, Wiles et al. (2010) uses the following matching procedure:

1. All firms in the same two-digit SIC code with market values of 50% to 150% of the focal firms are selected
2. From this list, the 10 firms with the most comparable book-to-market ratios are chosen to serve as the matched portfolio (the matched portfolio can have less than 10 firms).

Calculations:

$$AR_{it} = R_{it} - E(R_{it}|X_t)$$

Cumulative Abnormal Return (CAR):

$$CAR_{it} = \sum_{t=1}^{T}(R_{it} - E(R_{it}))$$

Buy-and-Hold Abnormal Return (BHAR)

$$BHAR_{t=1}^{T} = \Pi_{t=1}^{T}(1 + R_{it}) - \Pi_{t=1}^{T}(1 + E(R_{it}))$$

where as CAR is the arithmetic sum, BHAR is the geometric sum.

- In short-term event studies, differences between CAR and BHAR are often minimal. However, in long-term studies, this difference could significantly skew results. (Barber and Lyon, 1997) shows that while BHAR is usually slightly lower than annual CAR, but it dramatically surpasses CAR when annual BHAR exceeds 28%.

To calculate the long-run return ($\Pi_{t=1}^{T}(1+E(R_{it}))$) of the benchmark portfolio, we can:

1. **With annual rebalance**: In each period, each portfolio is re-balanced and then compound mean stock returns in a portfolio over a given period:

$$\Pi_{t=1}^{T}(1+E(R_{it})) = \Pi_{t}^{T}(1+\sum_{i=s}^{n_t} w_{it}R_{it})$$

where n_t is the number of firms in period t, and w_{it} is (1) $1/n_t$ or (2) value-weight of firm i in period t.

To avoid favoring recent events, in cross-sectional event studies, researchers usually treat all events equally when studying their impact on the stock market over time. This approach helps identify any abnormal changes in stock prices, especially when dealing with a series of unplanned events.

Potential problems:

- Solution first: Form benchmark portfolios that will never change constituent firms (Mitchell and Stafford, 2000), because of these problems:
  - Newly public companies often perform worse than a balanced market index (Ritter, 1991), and this, over time, might distort long-term return expectations due to the inclusion of these new companies (a phenomenon called "new listing bias" identified by Barber and Lyon (1997)).
  - Regularly rebalancing an equal-weight portfolio can lead to overestimated long-term returns and potentially skew buy-and-hold abnormal returns (BHARs) negatively due to constant selling of winning stocks and buying of underperformers (i.e., "rebalancing bias" (Barber and Lyon, 1997)).
  - Value-weight portfolios, which favor larger market cap stocks, can be viewed as an active investment strategy that keeps buying winning stocks and selling underperformers. Over time, this approach tends to positively distort BHARs.

2. **Without annual rebalance**: Compounding the returns of the securities comprising the portfolio, followed by calculating the average across all securities

$$\Pi_{t=s}^{T}(1+E(R_{it})) = \sum_{i=s}^{n_t}(w_{is}\Pi_{t=1}^{T}(1+R_{it}))$$

where t is the investment period, R_{it} is the return on security i, n_i is the number of securities, w_{it} is either $1/n_s$ or value-weight factor of security i at initial period s. This portfolio's profits come from a simple investment where all the included stocks are given equal importance, or weighted according to their market value, as they were in a specific past period (period s). This means that it doesn't consider any stocks that were listed after this period, nor does it adjust the portfolio each month. However, one problem with this method is that the value assigned to each stock, based on its market size, needs to be corrected. This is to make sure that recent stocks don't end up having too much influence.

Fortunately, on WRDS, it will give you all types of BHAR (2x2) (equal-weighted vs. value-weighted and with annual rebalance and without annual rebalance)

- "MINWIN" is the smallest number of months a company trades after an event to be included in the study.
- "MAXWIN" is the most months that the study considers in its calculations.
  - Companies aren't excluded if they have less than MAXWIN months, unless they also have fewer than MINWIN months.
- The term "MONTH" signifies chosen months (typically 12, 24, or 36) used to work out BHAR.
  - If monthly returns are missing during the set period, matching portfolio returns fill in the gaps.

32.5.2 Long-term Cumulative Abnormal Returns (LCARs)

Formula for LCARs during the $(1, T)$ postevent horizon (Sorescu et al., 2007)

$$LCAR_{pT} = \sum_{t=1}^{t=T}(R_{it} - R_{pt})$$

where R_{it} is the rate of return of stock i in month t

R_{pt} is the rate of return on the counterfactual portfolio in month t

32.5.3 Calendar-time Portfolio Abnormal Returns (CTARs)

This section follows strictly the procedure in (Wiles et al., 2010)

A portfolio for every day in calendar time (including all securities which experience an event that time).

For each portfolio, the securities and their returns are equally weighted

1. For all portfolios, the average abnormal return are calculated as

$$AAR_{Pt} = \frac{\sum_{i=1}^{S} AR_i}{S}$$

where

- S is the number of securities in portfolio P
- AR_i is the abnormal return for the stock i in the portfolio

2. For every portfolio P, a time series estimate of $\sigma(AAR_{Pt})$ is calculated for the preceding k days, assuming that the AAR_{Pt} are independent over time.
3. Each portfolio's average abnormal return is standardized

$$SAAR_{Pt} = \frac{AAR_{Pt}}{SD(AAR_{Pt})}$$

4. Average standardized residual across all portfolio's in calendar time

$$ASAAR = \frac{1}{n} \sum_{i=1}^{255} SAAR_{Pt} \times D_t$$

where

- $D_t = 1$ when there is at least one security in portfolio t
- $D_t = 0$ when there are no security in portfolio t
- n is the number of days in which the portfolio have at least one security $n = \sum_{i=1}^{255} D_t$

5. The cumulative average standardized average abnormal returns is

$$CASSAR_{S_1,S_2} = \sum_{i=S_1}^{S_2} ASAAR$$

If the ASAAR are independent over time, then standard deviation for the above estimate is $\sqrt{S_2 - S_1 + 1}$

then, the test statistics is

$$t = \frac{CASAAR_{S_1,S_2}}{\sqrt{S_2 - S_1 + 1}}$$

Limitations

- Cannot examine individual stock difference, can only see the difference at the portfolio level.
  - One can construct multiple portfolios (based on the metrics of interest) so that firms in the same portfolio shares that same characteristics. Then, one can compare the intercepts in each portfolio.
- Low power (Loughran and Ritter, 2000), type II error is likely.

32.6 Aggregation

32.6.1 Over Time

We calculate the cumulative abnormal (CAR) for the event windows

H_0: Standardized cumulative abnormal return for stock i is 0 (no effect of events on stock performance)

H_1: SCAR is not 0 (there is an effect of events on stock performance)

32.6.2 Across Firms + Over Time

Additional assumptions: Abnormal returns of different socks are uncorrelated (rather strong), but it's very valid if event windows for different stocks do not overlap. If the windows for different overlap, follow (Bernard, 1987) and (Schipper and Thompson, 1983, Schipper and Smith (1983))

H_0: The mean of the abnormal returns across all firms is 0 (no effect)

H_1: The mean of the abnormal returns across all firms is different form 0 (there is an effect)

Parametric (empirically either one works fine) (assume abnormal returns is normally distributed) :

1. Aggregate the CAR of all stocks (Use this if the true abnormal variance is greater for stocks with higher variance)
2. Aggregate the SCAR of all stocks (Use this if the true abnormal return is constant across all stocks)

Non-parametric (no parametric assumptions):

1. Sign test:
   - Assume both the abnormal returns and CAR to be independent across stocks

    - Assume 50% with positive abnormal returns and 50% with negative abnormal return
    - The null will be that there is a positive abnormal return correlated with the event (if you want the alternative to be there is a negative relationship)
    - With skewed distribution (likely in daily stock data), the size test is not trustworthy. Hence, rank test might be better
2. Rank test
    - Null: there is no abnormal return during the event window

32.7 Heterogeneity in the event effect

$$y = X\theta + \eta$$

where

- y = CAR
- X = Characteristics that lead to heterogeneity in the event effect (i.e., abnormal returns) (e.g., firm or event specific)
- η = error term

Note:

- In cases with selection bias (firm characteristics and investor anticipation of the event: larger firms might enjoy great positive effect of an event, and investors endogenously anticipate this effect and overvalue the stock), we have to use the White's t-statistics to have the lower bounds of the true significance of the estimates.
- This technique should be employed even if the average CAR is not significantly different from 0, especially when the CAR variance is high (Boyd et al., 2010)

32.7.1 Common variables in marketing

(Sorescu et al., 2017) Table 4

- Firm size is negatively correlated with abnormal return in finance (Sorescu et al., 2017), but mixed results in marketing.
- # of event occurrences
- R&D expenditure
- Advertising expense
- Marketing investment (SG&A)
- Industry concentration (HHI, # of competitors)

- Financial leverage
- Market share
- Market size (total sales volume within the firm's SIC code)
- marketing capability
- Book to market value
- ROA
- Free cash flow
- Sales growth
- Firm age

32.8 Expected Return Calculation

32.8.1 Statistical Models

- based on statistical assumptions about the behavior of returns (e..g, multivariate normality)
- we only need to assume stable distributions (Owen and Rabinovitch, 1983)

32.8.1.1 Constant Mean Return Model

The expected normal return is the mean of the real returns

$$Ra_{it} = R_{it} - \bar{R}_i$$

Assumption:

- returns revert to its mean (very questionable)

The basic mean returns model generally delivers similar findings to more complex models since the variance of abnormal returns is not decreased considerably (Brown and Warner, 1985)

32.8.1.2 Market Model

$$R_{it} = \alpha_i + \beta R_{mt} + \epsilon_{it}$$

where

- R_{it} = stock return i in period t
- R_{mt} = market return
- ϵ_{it} = zero mean ($E(e_{it}) = 0$) error term with its own variance σ^2

Notes:

- People typically use S&P 500, CRSP value-weighed or equal-weighted index as the market portfolio.
- When $\beta = 0$, the Market Model is the Constant Mean Return Model
- better fit of the market-model, the less variance in abnormal return, and the more easy to detect the event's effect
- recommend generalized method of moments to be robust against autocorrelation and heteroskedasticity

32.8.1.3 Fama-French Model

Please note that there is a difference between between just taking the return versus taking the excess return as the dependent variable.

The correct way is to use the excess return for firm and for market (Fama and French, 2010, p. 1917).

- α_i "is the average return left unexplained by the benchmark model" (i.e., abnormal return)

32.8.1.3.1 FF3 (Fama and French, 1993)

$$\begin{aligned} E(R_{it}|X_t) - r_{ft} = \alpha_i &+ \beta_{1i}(E(R_{mt}|X_t) - r_{ft}) \\ &+ b_{2i}SML_t + b_{3i}HML_t \end{aligned}$$

where

- r_{ft} risk-free rate (e.g., 3-month Treasury bill)
- R_{mt} is the market-rate (e.g., S&P 500)
- SML: returns on small (size) portfolio minus returns on big portfolio
- HML: returns on high (B/M) portfolio minus returns on low portfolio.

32.8.1.3.2 FF4 (Sorescu et al., 2017, p. 195) suggest the use of Market Model in marketing for short-term window and Fama-French Model for the long-term window (the statistical properties of this model have not been examined the the daily setting).

(Carhart, 1997)

$$\begin{aligned} E(R_{it}|X_t) - r_{ft} = \alpha_i &+ \beta_{1i}(E(R_{mt}|X_t) - r_{ft}) \\ &+ b_{2i}SML_t + b_{3i}HML_t + b_{4i}UMD_t \end{aligned}$$

where

- UMD_t is the momentum factor (difference between high and low prior return stock portfolios) in day t.

32.8.2 Economic Model

The only difference between CAPM and APT is that APT has multiple factors (including factors beyond the focal company)

Economic models put limits on a statistical model that come from assumed behavior that is derived from theory.

32.8.2.1 Capital Asset Pricing Model (CAPM)

$$E(R_i) = R_f + \beta_i(E(R_m) - R_f)$$

where

- $E(R_i)$ = expected firm return
- R_f = risk free rate
- $E(R_m - R_f)$ = market risk premium
- β_i = firm sensitivity

32.8.2.2 Arbitrage Pricing Theory (APT)

$$R = R_f + \Lambda f + \epsilon$$

where

- $\epsilon \sim N(0, \Psi)$
- Λ = factor loadings
- $f \sim N(\mu, \Omega)$ = general factor model
  - μ = expected risk premium vector
  - Ω = factor covariance matrix

32.9 Application

Packages:

- `eventstudies`
- `erer`
- `EventStudy`
- `AbnormalReturns`

- Event Study Tools
- estudy2
- `PerformanceAnalytics`

In practice, people usually sort portfolio because they are not sure whether the FF model is specified correctly.

Steps:

1. Sort all returns in CRSP into 10 deciles based on size.
2. In each decile, sort returns into 10 decides based on BM
3. Get the average return of the 100 portfolios for each period (i.e., expected returns of stocks given decile - characteristics)
4. For each stock in the event study: Compare the return of the stock to the corresponding portfolio based on size and BM.

Notes:

- Sorting produces outcomes that are often more conservative (e.g., FF abnormal returns can be greater than those that used sorting).
- If the results change when we do B/M first then size or vice versa, then the results are not robust (this extends to more than just two characteristics - e.g., momentum).

Examples:

Forestry:

- (Mei and Sun, 2008) M&A on financial performance (forest product)
- (Sun and Liao, 2011) litigation on firm values

```
library(erer)

# example by the package's author
data(daEsa)
hh <- evReturn(
    y = daEsa,        # dataset
    firm = "wpp",     # firm name
    y.date = "date", # date in y
    index = "sp500", # index
    est.win = 250,   # estimation window wedith in days
    digits = 3,
    event.date = 19990505, # firm event dates
    event.win = 5          # one-side event window wdith in days (default = 3, where 3
)
hh; plot(hh)
#>
#> === Regression coefficients by firm =========
```

```
#>   N firm event.date alpha.c alpha.e alpha.t alpha.p alpha.s beta.c beta.e
#> 1 1  wpp   19990505  -0.135   0.170  -0.795   0.428          0.665  0.123
#>   beta.t beta.p beta.s
#> 1  5.419  0.000    ***
#>
#> === Abnormal returns by date ================
#>    day Ait.wpp    HNt
#> 1   -5   4.564  4.564
#> 2   -4   0.534  5.098
#> 3   -3  -1.707  3.391
#> 4   -2   2.582  5.973
#> 5   -1  -0.942  5.031
#> 6    0  -3.247  1.784
#> 7    1  -0.646  1.138
#> 8    2  -2.071 -0.933
#> 9    3   0.368 -0.565
#> 10   4   4.141  3.576
#> 11   5   0.861  4.437
#>
#> === Average abnormal returns across firms ===
#>      name estimate error t.value p.value sig
#> 1 CiT.wpp    4.437 8.888   0.499   0.618
#> 2     GNT    4.437 8.888   0.499   0.618
```

Example by Ana Julia Akaishi Padula, Pedro Albuquerque (posted on LAMFO)

Example in `AbnormalReturns` package

32.9.1 Eventus

2 types of output:

1.
   - Using different estimation methods (e.g., market model to calendar-time approach)
   - Does not include event-specific returns. Hence, no regression later to determine variables that can affect abnormal stock returns.
2. Cross-sectional Analysis of Eventus: Event-specific abnormal returns (using monthly or data data) for cross-sectional analysis (under **Cross-Sectional Analysis** section)
   - Since it has the stock-specific abnormal returns, we can do regression on CARs later. But it only gives market-adjusted model. However, according to (Sorescu et al., 2017), they advocate for the use of market-adjusted model for the short-term only, and reserve the FF4 for the longer-term event studies using monthly daily.

32.9.1.1 Basic Event Study

1. Input a text file contains a firm identifier (e.g., PERMNO, CUSIP) and the event date
2. Choose market indices: equally weighted and the value weighted index (i.e., weighted by their market capitalization). And check Fama-French and Carhart factors.
3. Estimation options
   1. Estimation period: `ESTLEN = 100` is the convention so that the estimation is not impacted by outliers.
   2. Use "autodate" options: the first trading after the event date is used if the event falls on a weekend or holiday
4. Abnormal returns window: depends on the specific event
5. Choose test: either parametric (including Patell Standardized Residual (PSR)) or non-parametric

32.9.1.2 Cross-sectional Analysis of Eventus

Similar to the Basic Event Study, but now you can have event-specific abnormal returns.

32.9.2 Evenstudies

This package does not use the Fama-French model, only the market models.

This example is by the author of the package

```
library(eventstudies)
# firm and date data
data("SplitDates")
```

```r
head(SplitDates)

# stock price data
data("StockPriceReturns")
head(StockPriceReturns)
class(StockPriceReturns)

es <-
    eventstudy(
        firm.returns = StockPriceReturns,
        event.list = SplitDates,
        event.window = 5,
        type = "None",
        to.remap = TRUE,
        remap = "cumsum",
        inference = TRUE,
        inference.strategy = "bootstrap"
    )

plot(es)
```

32.9.3 EventStudy

You have to pay for the API key. (It's $10/month).

```r
library(EventStudy)
```

Example by the authors of the package

Data Prep

```r
library(tidyquant)
library(tidyverse)
library(readr)
```

Reference market in Germany is DAX

```r
# Index Data
# indexName <- c("DAX")

indexData <- tq_get("^GDAXI", from = "2014-05-01", to = "2015-12-31") %>%
    mutate(date = format(date, "%d.%m.%Y")) %>%
    mutate(symbol = "DAX")

head(indexData)
```

Create files

- 01_RequestFile.csv
- 02_FirmData.csv
- 03_MarketData.csv

Calculating abnormal returns

```
# get & set parameters for abnormal return Event Study
# we use a garch model and csv as return
# Attention: fitting a GARCH(1, 1) model is compute intensive
esaParams <- EventStudy::ARCApplicationInput$new()
esaParams$setResultFileType("csv")
esaParams$setBenchmarkModel("garch")


dataFiles <-
    c(
        "request_file" = file.path(getwd(), "data", "EventStudy", "01_requestFile.csv")
        "firm_data"    = file.path(getwd(), "data", "EventStudy", "02_firmDataPrice.cs
        "market_data"  = file.path(getwd(), "data", "EventStudy", "03_marketDataPrice.
    )

# check data files, you can do it also in our R6 class
EventStudy::checkFiles(dataFiles)
```

```
arEventStudy <- estSetup$performEventStudy(estParams     = esaParams,
                                           dataFiles     = dataFiles,
                                           downloadFiles = T)
```

```
library(EventStudy)

apiUrl <- "https://api.eventstudytools.com"
Sys.setenv(EventStudyapiKey = "")

# The URL is already set by default
options(EventStudy.URL = apiUrl)
options(EventStudy.KEY = Sys.getenv("EventStudyapiKey"))

# use EventStudy estAPIKey function
estAPIKey(Sys.getenv("EventStudyapiKey"))

# initialize object
estSetup <- EventStudyAPI$new()
estSetup$authentication(apiKey = Sys.getenv("EventStudyapiKey"))
```

Chapter 33

Instrumental Variables

Similar to RCT, we try to introduce randomization (random assignment to treatment) to our treatment variable by using only variation in the instrument.

Logic of using an instrument:

- Use only exogenous variation to see the variation in treatment (try to exclude all endogenous variation in the treatment)
- Use only exogenous variation to see the variation in outcome (try to exclude all endogenous variation in the outcome)
- See the relationship between treatment and outcome in terms of residual variations that are exogenous to omitted variables.

Notes:

- Instruments can be used to remove attenuation bias in errors-in-variables.
- Be careful with the F-test and standard errors when you do 2SLS by hand (you need to correct them).
- Repeated use of related IVs across different studies can collectively invalidate these instruments, primarily through the violation of the exclusion restriction (Gallen, 2020). One needs to test for invalid instruments (Hausman-like test).
  - Mellon (2023) shows the widespread use of weather as an instrument in social sciences (289 studies linking weather to 195 variables) demonstrates significant exclusion violations that can overturn many IV results.
- For [Zero-valued Outcomes], we can't directly interpret the treatment coefficient of log-transformed outcome regression as percentage change (Chen and Roth, 2023). We have to distinguish the treatment effect on the

intensive (outcome: 10 to 11) vs. extensive margins (outcome: 0 to 1), and we can't readily interpret the treatment coefficient of log-transformed outcome regression as percentage change. To have percentage change interpretation, we can either do:

- Proportional LATE: estimate $\theta_{ATE\%}$ for those who are compliers under the instrument. To estimate proportional LATE,

  1. Regress $Y_i = \beta D_i + X_i + \epsilon_i$ using 2SLS with an instrument on D_i, where β is interpreted as the LATE in levels of the control group's mean for compliers.

  2. Get estimate of the control complier mean by regressing with same 2SLS regression (Abadie et al., 2002) where the final outcome is $-(D_i - 1)Y_i$, we refer to the new new estimated effect of D_i as β_{cc}

  3. The $\theta_{ATE\%}$ for compliers that are induced by the instrument is $\widehat{\beta}/\widehat{\beta}_{cc}$, where it can be interpreted directly as the percentage change for compliers that are induced by the instrument under treatment as compared to under control.

  4. SE can be obtained by non-parametric bootstrap.

  5. For specific case that the instrument is binary, θ of the intensive margin for compliers can be directly obtained by Poisson IV regression (`ivpoisson` in Stata).

- Lee (2009) bounds: we can get bounds for the average treatment effect in logs for compliers who have positive outcome regardless of treatment status (i.e., intensive-margin effect). This requires a monotonicity assumption for compliers where they should still have positive outcome regardless of treatment status.

Notes on First-stage:

- Always use the OLS regression in the first stage (regardless of the type of endogenous variables - e.g., continuous or discreet) (suggested by (Angrist and Pischke, 2009). Estimates of IV can still be consistent regardless of the form of the endogenous variables (discreet vs. continuous).

  - Alternatively, we could use "biprobit" model, but this is applicable only in cases where you have both dependent and endogenous variables to be binary.

- If you still want to continue and use logit or probit models for the first stage when you have binary variables, you have a "forbidden regression" (also 1, 2) (i.e., an incorrect extension of 2SLS to a nonlinear case).

There are several ways to understand this problem:

1. **Identification strategy**: The identification strategy in instrumental variables analysis relies on the fact that the instrumental variable affects the outcome variable only through its effect on the endogenous variable. However, when the endogenous variable is binary, the relationship between the instrumental variable and the endogenous variable is not continuous. This means that the instrumental variable can only affect the endogenous variable in discrete jumps, rather than through a continuous change. As a result, the identification of the causal effect of the endogenous variable on the outcome variable may not be possible with probit or logit regression in the first stage.

2. **Model assumptions**: Both models assume that the error term has a specific distribution (normal or logistic), and that the probability of the binary outcome is a function of the linear combination of the regressors.

   When the endogenous variable is binary, however, the distribution of the error term is not specified, as there is no continuous relationship between the endogenous variable and the outcome variable. This means that the assumptions of the probit and logit models may not hold, and the resulting estimates may not be reliable or interpretable.

3. **Issue of weak instruments**: When the instrument is weak, the variance of the inverse Mills ratio (which is used to correct for endogeneity in instrumental variables analysis) can be very large. In the case of binary endogenous variables, the inverse Mills ratio cannot be consistently estimated using probit or logit regression, and this can lead to biased and inconsistent estimates of the causal effect of the endogenous variable on the outcome variable.

Problems with weak instruments (Bound et al., 1995):

- Weak instrumental variables can produce (finite-sample) **biased** and **inconsistent** estimates of the causal effect of an endogenous variable on an outcome variable (even in the presence of large sample size)
- In a finite sample, instrumental variables (IV) estimates can be biased in the same direction as ordinary least squares (OLS) estimates. Additionally, the bias of IV estimates approaches that of OLS estimates as the correlation (R2) between the instruments and the endogenous explanatory variable approaches zero. This means that when the correlation between the instruments and the endogenous variable is weak, the bias of the IV estimates can be similar to that of the OLS estimates.
- Weak instruments are problematic because they do not have enough variation to fully capture the variation in the endogenous variable, leading to measurement error and other sources of noise in the estimates.
- Using weak instruments can produce large standard errors and low t-ratio. And when the feedback (reverse causality) is strong, the bias in IV is even greater than that of OLS (Nelson and Startz, 1988).

- Using lagged dependent variables as instruments for current values depends on serial correlations, typically low (Nelson and Startz, 1988).
- Using multiple covariates to artificially increase the first-stage R^2 does not solve this weak instrument problem (Nelson and Startz, 1988).
- Solutions:
  - use of multiple instruments
  - use of instrumental variables with higher correlation
  - use of alternative estimation methods such as limited information maximum likelihood (LIML) or two-stage least squares (2SLS) with heteroscedasticity-robust standard errors.

Instrument Validity:

1. Random assignment (Exogeneity Assumption).
2. Any effect of the instrument on the outcome must be through the endogenous variable (Relevance Assumption).

33.1 Framework

- $D_i \sim Bern$ Dummy Treatment
- Y_{0i}, Y_{1i} potential outcomes
- $Y_i = Y_{0i} + (Y_{1i} - Y_{0i})D_i$ observed outcome
- $Z_i \perp Y_{0i}, Y_{1i}$ Instrumental variables (and also correlate with D_i)

Under constant-effects and linear ($Y_{1i} - Y_{0i}$ are the same for everyone)

$$\begin{aligned} Y_{0i} &= \alpha + \eta_i \\ Y_{1i} - Y_{0i} &= \rho \\ Y_i &= Y_{0i} + D_i(Y_{1i} - Y_{0i}) \\ &= \alpha + \eta_i + D_i\rho \\ &= \alpha + \rho D_i + \eta_i \end{aligned}$$

where

- η_i is individual differences
- ρ is the difference between treated outcome and untreated outcome. Here we assume they are constant for everyone

However, we have a problem with OLS because D_i is correlated with η_i for each unit

But Z_i can come to the rescue, the causal estimate can be written as

$$\begin{aligned}\rho &= \frac{Cov(Y_i, Z_i)}{Cov(D_i, Z_i)} \\ &= \frac{Cov(Y_i, Z_i)/V(Z_i)}{Cov(D_i, Z_i)/V(Z_i)} = \frac{Reducedform}{First-stage} \\ &= \frac{E[Y_i|Z_i = 1] - E[Y_i|Z_i = 0]}{E[D_i|Z_i = 1] - E[D_i|Z_i = 0]}\end{aligned}$$

Under heterogeneous treatment effect ($Y_{1i} - Y_{0i}$ are different for everyone) with LATE framework

$Y_i(d, z)$ denotes the potential outcome for unit i with treatment $D_i = d$ and instrument $Z_i = z$

Observed treatment status

$$D_i = D_{0i} + Z_i(D_{1i} - D_{0i})$$

where

- D_{1i} is treatment status of unit i when $z_i = 1$
- D_{0i} is treatment status of unit i when $z_i = 0$
- $D_{1i} - D_{0i}$ is the causal effect of Z_i on D_i

Assumptions

- ***Independence***: The instrument is randomly assigned (i.e., independent of potential outcomes and potential treatments)
  - $[\{Y_i(d, z); \forall d, z\}, D_{1i}, D_{0i}]\Pi Z_i$
  - This assumption let the first-stage equation be the average causal effect of Z_i on D_i

$$\begin{aligned}E[D_i|Z_i = 1] - E[D_i|Z_i = 0] &= E[D_{1i}|Z_i = 1] - E[D_{0i}|Z_i = 0] \\ &= E[D_{1i} - D_{0i}]\end{aligned}$$

  - This assumption also is sufficient for a causal interpretation of the reduced form, where we see the effect of the instrument on the outcome.

$$E[Y_i|Z_i = 1] - E[Y_i|Z_i = 0] = E[Y_i(D_{1i}, Z_i = 1) - Y_i(D_{0i}, Z_i = 0)]$$

- ***Exclusion*** (i.e., existence of instruments (Imbens and Angrist, 1994)
  - The treatment D_i fully mediates the effect of Z_i on Y_i

$$Y_{1i} = Y_i(1, 1) = Y_i(1, 0)Y_{0i} = Y_i(0, 1) = Y_i(0, 0)$$

  - With this assumption, the observed outcome Y_i can be thought of as (assume Y_{1i}, Y_{0i} already satisfy the independence assumption)

$$\begin{aligned} Y_i &= Y_i(0, Z_i) + [Y_i(1, Z_i) - Y_i(0, Z_i)]D_i \\ &= Y_{0i} + (Y_{1i} - Y_{0i})D_i \end{aligned}$$

  - This assumption let us go from reduced-form causal effects to treatment effects (Angrist and Imbens, 1995)

- ***Monotonicity***: $D_{1i} > D_{0i} \forall i$
  - With this assumption, we have $E[D_{1i} - D_{0i}] = P[D_{1i} > D_{0i}]$
  - This assumption lets us assume that there is a first stage, in which we examine the proportion of the population that D_i is driven by Z_i
  - This assumption is used to solve to problem of the shifts between participation status back to non-participation status.
    * Alternatively, one can solve the same problem by assuming constant (homogeneous) treatment effect (Imbens and Angrist, 1994), but this is rather restrictive.
    * A third solution is the assumption that there exists a value of the instrument, where the probability of participation conditional on that value is 0 (Heckman, 1990, Angrist and Imbens (1991)).

With these three assumptions, we have the LATE theorem (Angrist and Pischke, 2009, 4.4.1)

$$\frac{E[Y_i|Z_i = 1] - E[Y_i|Z_i = 0]}{E[D_i|Z_i = 1] - E[D_i|Z_i = 0]} = E[Y_{1i} - Y_{0i}|D_{1i} > D_{0i}]$$

LATE assumptions allow us to go back to the types of subjects we have in Causal Inference

- Switchers:
  - Compliers: $D_{1i} > D_{0i}$
- Non-switchers:
  - Always-takers: $D_{1i} = D_{0i} = 1$
  - Never-takers: $D_{1i} = D_{0i} = 0$

Instrumental Variables can't say anything about non-switchers because treatment status D_i has no effects on them (similar to fixed effects models).

When all groups are the same, we come back to the constant-effects world.

Treatment effects on the treated is a weighted average of always-takers and compliers.

In the special case of IV in randomized trials, we have a compliance problem (when compliance is voluntary), where those in the treated will not always take the treatment (i.e., might be selection bias).

- **Intention-to-treat analysis** is valid, but contaminated by non-compliance
- IV in this case (Z_i = random assignment to the treatment; D_i = whether the unit actually received/took the treatment) can solve this problem.
- Under certain assumptions (i.e., SUTVA, random assignment, exclusion restriction, no defiers, and monotinicity), this analysis can give causal interpreation of LATE because it's the average causal effect for the compliers only.
  - Without these assumptions, it's a ratio of intention-to-treat.
- Without always-takers in this case, LATE = Treatment effects on the treated

See proof Bloom (1984) and examples Bloom et al. (1997) and Sherman and Berk (1984)

$$\frac{E[Y_i|Z_i=1]-E[Y_i|Z_i=0]}{E[D_i|Z_i=1]} = \frac{\text{Intention-to-treat effect}}{\text{Compliance rate}} = E[Y_{1i}-Y_{0i}|D_i=1]$$

33.2 Estimation

33.2.1 2SLS Estimation

A special case of IV-GMM

Examples by authors of `fixest` package

```
library(fixest)
base = iris
names(base) = c("y", "x1", "x_endo_1", "x_inst_1", "fe")
set.seed(2)
base$x_inst_2 = 0.2 * base$y + 0.2 * base$x_endo_1 + rnorm(150, sd = 0.5)
base$x_endo_2 = 0.2 * base$y - 0.2 * base$x_inst_1 + rnorm(150, sd = 0.5)

# est_iv = feols(y ~ x1 | x_endo_1  ~ x_inst_1 , base)
est_iv = feols(y ~ x1 | x_endo_1 + x_endo_2 ~ x_inst_1 + x_inst_2, base)
est_iv
#> TSLS estimation - Dep. Var.: y
#>                   Endo.    : x_endo_1, x_endo_2
#>                   Instr.   : x_inst_1, x_inst_2
#> Second stage: Dep. Var.: y
#> Observations: 150
```

```
#> Standard-errors: IID
#>              Estimate Std. Error  t value   Pr(>|t|)
#> (Intercept)  1.831380   0.411435  4.45121 1.6844e-05 ***
#> fit_x_endo_1 0.444982   0.022086 20.14744  < 2.2e-16 ***
#> fit_x_endo_2 0.639916   0.307376  2.08186 3.9100e-02 *
#> x1           0.565095   0.084715  6.67051 4.9180e-10 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> RMSE: 0.398842   Adj. R2: 0.761653
#> F-test (1st stage), x_endo_1: stat = 903.2    , p < 2.2e-16 , on 2 and 146 DoF.
#> F-test (1st stage), x_endo_2: stat =   3.25828, p = 0.041268, on 2 and 146 DoF.
#>                   Wu-Hausman: stat =   6.79183, p = 0.001518, on 2 and 144 DoF.
```

Default statistics

1. F-test first-stage (weak instrument test)
2. Wu-Hausman endogeneity test
3. Over-identifying restriction (Sargan) J-test

```r
fitstat(
    est_iv,
    type = c(
        "n", "ll", "aic", "bic", "rmse", # ll means log-likelihood

        "my", # mean dependent var

        "g", # degrees of freedom used to compute the t-test

        "r2", "ar2", "wr2", "awr2", "pr2", "apr2", "wpr2", "awpr2",

        "theta", # over-dispersion parameter in Negative Binomial models

        "f", "wf", # F-tests of nullity of the coefficients

        "wald", # Wald test of joint nullity of the coefficients

        "ivf",

        "ivf1",

        "ivf2",

        "ivfall",

        "ivwald", "ivwald1", "ivwald2", "ivwaldall",
```

```
        "cd"

        # "kpr"


        ),
    cluster = 'fe'
)
#>                 Observations: 150
#>               Log-Likelihood: -75.0
#>                          AIC: 157.9
#>                          BIC: 170.0
#>                         RMSE: 0.398842
#>               Dep. Var. mean: 5.84333
#>                            G: 3
#>                           R2: 0.766452
#>                      Adj. R2: 0.761653
#>                    Within R2: NA
#>                         awr2: NA
#>                    Pseudo R2: 0.592684
#>               Adj. Pseudo R2: 0.576383
#>             Within Pseudo R2: NA
#>                        awpr2: NA
#>              Over-dispersion: NA
#>                       F-test: stat =       1.80769, p = 0.375558, on 3 and 2 DoF.
#>           F-test (projected): NA
#>         Wald (joint nullity): stat = 539,363.2    , p < 2.2e-16 , on 3 and 146 DoF, VCOV: Clus
#> F-test (1st stage), x_endo_1: stat =     903.2    , p < 2.2e-16 , on 2 and 146 DoF.
#> F-test (1st stage), x_endo_2: stat =       3.25828, p = 0.041268, on 2 and 146 DoF.
#>           F-test (2nd stage): stat =     194.2    , p < 2.2e-16 , on 2 and 146 DoF.
#>             F-test (IV only): stat =     194.2    , p < 2.2e-16 , on 2 and 146 DoF.
#> Wald (1st stage), x_endo_1  : stat =   1,482.6    , p < 2.2e-16 , on 2 and 146 DoF, VCOV: Clus
#> Wald (1st stage), x_endo_2  : stat =       2.22157, p = 0.112092, on 2 and 146 DoF, VCOV: Clus
#>             Wald (2nd stage): stat = 539,363.2    , p < 2.2e-16 , on 2 and 146 DoF, VCOV: Clus
#>               Wald (IV only): stat = 539,363.2    , p < 2.2e-16 , on 2 and 146 DoF, VCOV: Clus
#>                 Cragg-Donald: 3.11162
```

To set default printing

```
# always add second-stage Wald test
setFixest_print(fitstat = ~ . + ivwald2)
est_iv
```

To see results from different stages

```r
# first-stage
summary(est_iv, stage = 1)

# second-stage
summary(est_iv, stage = 2)

# both stages
etable(summary(est_iv, stage = 1:2), fitstat = ~ . + ivfall + ivwaldall.p)
etable(summary(est_iv, stage = 2:1), fitstat = ~ . + ivfall + ivwaldall.p)
# .p means p-value, not statistic
# `all` means IV only
```

33.2.2 IV-GMM

This is a more general framework.

- 2SLS Estimation is a special case of IV-GMM estimator

$$Y = X\beta + u, u \sim (0, \Omega)$$

where

- X is a matrix of endogenous variables $(N \times k)$

We will use a matrix of instruments X where it has $N \times l$ dimensions (where $l \geq k$)

Then, we can have a set of l moments:

$$g_i(\beta) = Z_i' u_i = Z_i'(Y_i - X_i\beta)$$

where

- $i \in (1, N)$

Each l moment equation is a sample moment, which can be estimated by averaging over N

$$\bar{g}(\beta) = \frac{1}{N}\sum_{i=1}^{N} Z_i(Y_i - X_i\beta) = \frac{1}{N}Z'u$$

GMM then estimate β so that $\bar{g}(\hat{\beta}_{GMM}) = 0$

When $l = k$ there is a unique solution to this system of equations (and equivalent to the IV estimator)

$$\hat{\beta}_{IV} = (Z'X)^{-1}Z'Y$$

When $l > k$, we have a set of k instruments

$$\hat{X} = Z(Z'Z)^{-1}Z'X = P_Z X$$

then we can use the 2SLS estimator

$$\begin{aligned}\hat{\beta}_{2SLS} &= (\hat{X}'X)^{-1}\hat{X}'Y \\ &= (X'P_Z X)^{-1}X'P_Z Y\end{aligned}$$

Differences between 2SLS and IV-GMM:

- In the 2SLS method, when there are more instruments available than what is actually needed for the estimation, to address this, a matrix is created that only includes the necessary instruments, which simplifies the calculation.
- The IV-GMM method uses all the available instruments, but applies a weighting system to prioritize the instruments that are most relevant. This approach is useful when there are more instruments than necessary, which can make the calculation more complex. The IV-GMM method uses a criterion function to weight the estimates and improve their accuracy.
- **In short, always use IV-GMM when you have overid problems**

GMM estimator minimizes

$$J(\hat{\beta}_{GMM}) = N\bar{g}(\hat{\beta}_{GMM})'W\bar{g}(\hat{\beta}_{GMM})$$

where W is a symmetric weighting matrix $l \times l$

For an overid equation, solving the set of FOCs for the IV-GMM estimator, we should have

$$\hat{\beta}_{GMM} = (X'ZWZ'X)^{-1}X'ZWZ'Y$$

which is identical for all W matrices. The optimal $W = S^{-1}$ (Hansen, 1982) where S is the covariance matrix of the moment conditions to produce the most efficient estimator:

$$S = E[Z'uu'Z] = \lim_{N\to\infty} N^{-1}[Z'\Omega Z]$$

With a consistent estimator of S from the 2SLS residuals, the feasible IV-GMM estimator can be defined as

$$\hat{\beta}_{FEGMM} = (X'Z\hat{S}^{-1}Z'X)^{-1}X'Z\hat{S}^{-1}Z'Y$$

In cases where Ω (i.e., the vcov of the error process u) satisfy all classical assumptions

1. IID
2. $S = \sigma_u^2 I_N$
3. The optimal weighting matrix is proportional to the identity matrix

Then, IV-GMM estimator is the standard IV (or 2SLS) estimator.

For IV-GMM, you also have an additional test of overid restrictions: GMM distance (also known as Hayashi C statistic)

To account for clustering, one can use code provided by this blog

33.3 Inference

Under just-identified instrument variable model, we have

$$Y = \beta X + u$$

where $corr(u, Z) = 0$ (relevant assumption) and $corr(Z, X) \neq 0$ (exogenous assumption)

The t-ratio approach to construct the 95 CIs is

$$\hat{\beta} \pm 1.96\sqrt{\hat{V}_N(\hat{\beta})}$$

But this is wrong, and has been long recognized by those who understand the "weak instruments" problem (Staiger and Stock, 1997, Dufour (1997))

To test the null hypothesis of $\beta = \beta_0$ (Lee et al., 2022)

$$\frac{(\hat{\beta} - \beta_0)^2}{\hat{V}_N(\hat{\beta})} = \hat{t}^2 = \hat{t}_{AR}^2 \times \frac{1}{1 - \hat{\rho}\frac{\hat{t}_{AR}}{\hat{f}} + \frac{\hat{t}_{AR}^2}{\hat{f}^2}}$$

where $\hat{t}_{AR}^2 \sim \chi^2(1)$ (even with weak instruments) (Anderson and Rubin, 1949)

$$\hat{t}_{AR} = \frac{\hat{\pi}(\hat{\beta} - \beta_0)}{\sqrt{\hat{V}_N(\hat{\pi}(\hat{\beta} - \beta_0))}} \sim N(0, 1)$$

where

- $\hat{f} = \frac{\hat{\pi}}{\sqrt{\hat{V}_N(\hat{\pi})}} \sim N$

- $\hat{\pi}$ = 1st-stage coefficient
- $\hat{\rho} = COV(Zv, Zu)$ = correlation between the 1st-stage residual and an estimate of u

Even in large samples, $\hat{t}^2 \neq \hat{t}^2_{AR}$ because the right-hand term does not have a degenerate distribution. Thus, the normal t critical values wouldn't work.

The t-ratios does not match that of standard normal, but it matches the proposed density by Staiger and Stock (1997) and Stock and Yogo (2005b) .

The deviation between $\hat{t}^2, \hat{t}^2_{AR}$ depends on

1. π (i.e., correlation between the instrument and the endogenous variable)
2. $E(F)$ (i.e., strength of the first-stage)
3. Magnitude of $|\rho|$ (i.e., degree of endogeneity)

Hence, we can think of several scenarios:

1. **Worst case**: Very weak first stage ($\pi = 0$) and high degree of endogeneity ($|\rho| = 1$).

- The interval $\hat{\beta} \pm 1.96 \times SE$ does not contain the true parameter β.
- A 5 percent significance test under these conditions will incorrectly reject the null hypothesis ($\beta = \beta_0$) 100% of the time.

2. **Best case**: No endogeneity ($\rho = 0$) or very large $\hat{f}$ (very strong first-stage)

- The interval $\hat{\beta} \pm 1.96 \times SD$ accurately contains β at least 95% of the time.

3. **Intermediate case**: The performance of the interval lies between the two extremes.

Solutions: To have valid inference of $\hat{\beta} \pm 1.96 \times SE$ using t-ratio ($\hat{t}^2 \approx \hat{t}^2_{AR}$), we can either

1. Assume our problem away
    1. Assume $E(F) > 142.6$ (Lee et al., 2022) (Not much of an assumption since we can observe first-stage F-stat empirically).
    2. Assume $|\rho| < 0.565$ (Angrist and Kolesár, 2024, Lee et al. (2022)), but this defeats our motivation to use IV in the first place because we think there is a strong endogeneity bias, that's why we are trying to correct for it (circular argument).
2. Deal with it head on
    1. AR approach (Anderson and Rubin, 1949)
    2. tF Procedure (Lee et al., 2022)
    3. AK approach (Angrist and Kolesár, 2023)

Common Practices & Challenges:

- The t-ratio test is preferred by many researchers but has its pitfalls:

  - Known to over-reject (equivalently, under-cover confidence intervals), especially with weak instruments (Lee et al., 2022, Nelson and Startz (1990), Bound et al. (1995), Dufour (1997)).

- To address this:
  - The first-stage F-statistic is used as an indicator of weak instruments.
  - (Staiger and Stock, 1997, Stock and Yogo (2005b)) provided a framework to understand and correct these **distortions**.

Misinterpretations:

- Common errors in application:
  - Using a rule-of-thumb F-stat threshold of 10 instead of referring to Stock and Yogo (2005b).
  - Mislabeling intervals such as $\hat{\beta} \pm 1.96 \times \hat{se}(\hat{\beta})$ as 95% confidence intervals (when passed the $F > 10$ rule of thumb). Staiger and Stock (1997) clarified that such intervals actually represent 85% confidence when using $F > 16.38$ from Stock and Yogo (2005b)
- Pretesting for weak instruments might exacerbate over-rejection of the t-ratio test mentioned above (Hall et al., 1996).
- Selective model specification (i.e., dropping certain specification) based on F-statistics also leads to significant **distortions** (Andrews et al., 2019).

33.3.1 AR approach

Validity of Anderson-Rubin Test (notated as AR) (Anderson and Rubin, 1949):

- Gives accurate results even under non-normal and homoskedastic errors (Staiger and Stock, 1997).
- Maintains validity across diverse error structures (Stock and Wright, 2000).
- Minimizes type II error among several alternative tests, in cases of:
  - Homoskedastic errors (Andrews et al., 2006, Moreira (2009)).
  - Generalized for heteroskedastic, clustered, and autocorrelated errors (Moreira and Moreira, 2019).

```r
library(ivDiag)

# AR test (robust to weak instruments)
# example by the package's authors
ivDiag::AR_test(
    data = rueda,
```

```
    Y = "e_vote_buying",
    # treatment
    D = "lm_pob_mesa",
    # instruments
    Z = "lz_pob_mesa_f",
    controls = c("lpopulation", "lpotencial"),
    cl = "muni_code",
    CI = FALSE
)
#> $Fstat
#>         F       df1       df2         p
#>   50.5097    1.0000 4350.0000    0.0000

g <- ivDiag::ivDiag(
    data = rueda,
    Y = "e_vote_buying",
    D = "lm_pob_mesa",
    Z = "lz_pob_mesa_f",
    controls = c("lpopulation", "lpotencial"),
    cl = "muni_code",
    cores = 4,
    bootstrap = FALSE
)
g$AR
#> $Fstat
#>         F       df1       df2         p
#>   50.5097    1.0000 4350.0000    0.0000
#>
#> $ci.print
#> [1] "[-1.2545, -0.7156]"
#>
#> $ci
#> [1] -1.2545169 -0.7155854
#>
#> $bounded
#> [1] TRUE
ivDiag::plot_coef(g)
```

33.3.2 tF Procedure

Lee et al. (2022) propose a new method that is aligned better with traditional econometric training than AR, where it is called the tF procedure. It incorporates both the 1st-stage F-stat and the 2SLS t-value. This method is applicable to single instrumental variable (i.e., just-identified model), including

- Randomized trials with imperfect compliance (Imbens and Angrist, 1994).
- Fuzzy Regression Discontinuity designs (Lee and Lemieux, 2010).
- Fuzzy regression kink designs (Card et al., 2015).

See Andrews et al. (2019) for a comparison between AR approach and tF Procedure.

tF Procedure:

- Adjusts the t-ratio based on the first-stage F-statistic.
- Rather than a fixed pretesting threshold, it applies an adjustment factor to 2SLS standard errors.
- Adjustment factors are provided for 95% and 99% confidence levels.

Advantages of the tF Procedure:

1. **Smooth Adjustment**:
    - Gives usable finite confidence intervals for smaller F statistic values.
    - 95% confidence is applicable for $F > 3.84$, aligning with AR's bounded 95% confidence intervals.

2. **Clear Confidence Levels**:
   - These levels incorporate effects of basing inference on the first-stage F.
   - Mirrors AR or other zero distortion procedures.
3. **Robustness**:
   - Robust against common error structures (e.g., heteroskedasticity or clustering and/or autocorrelated errors).
   - No further adjustments are necessary as long as robust variance estimators are consistently used (same robust variance estimator used for the 1st-stage as for the IV estimate).
4. **Comparison to AR**:
   - Surprisingly, with $F > 3.84$, AR's expected interval length is infinite, while tF's is finite (i.e., better).
5. **Applicability**:
   - The tF adjustment can re-evaluate published studies if the first-stage F-statistic is available.
   - Original data access is not needed.

Impacts in Applied Research:

- Lee et al. (2022) examined recent single-instrument specification studies from the American Economic Review (AER).
- Observations:
  - For at least 25% of the studied specifications, using tF increased confidence interval lengths by:
    - 49% (5% significance level).
    - 136% (1% significance level).
  - For specifications with $F > 10$ and $t > 1.96$, about 25% became statistically insignificant at the 5% level when adjusted using tF.
  - Conclusion: tF adjustments could greatly influence inferences in research employing t-ratio inferences.

Notation

- $Y = X\beta + W\gamma + u$
- $X = Z\pi + W\xi + \nu$

where

- W: Additional covariates, possibly including an intercept term.
- X: variable of interest

- Z: instruments

Key Statistics:

- t-ratio for the instrumental variable estimator: $\hat{t} = \frac{\hat{\beta}-\beta_0}{\sqrt{\hat{V}_N(\hat{\beta})}}$
- t-ratio for the first-stage coefficient: $\hat{f} = \frac{\hat{\pi}}{\sqrt{\hat{V}_N(\hat{\pi})}}$
- $\hat{F} = \hat{f}^2$

where

- $\hat{\beta}$: Instrumental variable estimator.
- $\hat{V}_N(\hat{\beta})$: Estimated variance of $\hat{\beta}$, possibly robust to deal with non-iid errors.
- $\hat{t}$: t-ratio under the null hypothesis.
- $\hat{f}$: t-ratio under the null hypothesis of $\pi = 0$.

Traditional t Inference:

- In large samples, $\hat{t}^2 \to^d t^2$
- Standard normal critical values are ± 1.96 for 5% significance level testing.

Distortions in Inference in the case of IV:

- Use of a standard normal can lead to distorted inferences even in large samples.
  - Despite large samples, t-distribution might not be normal.
- But magnitude of this distortion can be quantified.
  - Stock and Yogo (2005b) provides a formula for Wald test statistics using 2SLS.
  - t^2 formula allows for quantification of inference distortions.
  - In the just-identified case with one endogenous regressor $t^2 = f + t_{AR} + \rho f t_{AR}$ (Stock and Yogo, 2005b)
    * $\hat{f} \to^d f$ and $\bar{f} = \frac{\pi}{\sqrt{\frac{1}{N}AV(\hat{\pi})}}$ and $AV(\hat{\pi})$ is the asymptotic variance of $\hat{\pi}$
    * t_{AR} is a standard normal with $AR = t_{AR}^2$
    * ρ (degree of endogeneity) is the correlation of Zu and $Z\nu$ (when data are homoskedastic, ρ is the correlation between u and ν)

Implications of t^2 formula:

- Varies rejection rates depending on ρ value.
  - $\rho \in (0, 0.5]$ (low) the t-ratio rejects at a probability below the nominal 0.05 rate
  - $\rho = 0.8$ (high) the rejection rate can be 0.13
- In short, incorrect test size when relying solely on t^2 (based on traditional econometric understanding)

To correct for this, one can

1. Estimate the usually 2SLS standard errors
2. Multiply the SE by the adjustment factor based on the observed first-stage $\hat{F}$ stat
3. One can go back to the traditional hypothesis by using either the t-ratio of confidence intervals

Lee et al. (2022) call this adjusted SE as "0.05 tF SE".

```
library(ivDiag)
g <- ivDiag::ivDiag(
    data = rueda,
    Y = "e_vote_buying",
    D = "lm_pob_mesa",
    Z = "lz_pob_mesa_f",
    controls = c("lpopulation", "lpotencial"),
    cl = "muni_code",
    cores = 4,
    bootstrap = FALSE
)
g$tF
#>         F        cF      Coef        SE         t    CI2.5%   CI97.5%   p-value
#> 8598.3264    1.9600   -0.9835    0.1540   -6.3872   -1.2853   -0.6817    0.0000
```

```
# example in fixest package
library(fixest)
library(tidyverse)
base = iris
names(base) = c("y", "x1", "x_endo_1", "x_inst_1", "fe")
set.seed(2)
base$x_inst_2 = 0.2 * base$y + 0.2 * base$x_endo_1 + rnorm(150, sd = 0.5)
base$x_endo_2 = 0.2 * base$y - 0.2 * base$x_inst_1 + rnorm(150, sd = 0.5)

est_iv = feols(y ~ x1 | x_endo_1 + x_endo_2 ~ x_inst_1 + x_inst_2, base)
est_iv
#> TSLS estimation - Dep. Var.: y
#>                   Endo.    : x_endo_1, x_endo_2
#>                   Instr.   : x_inst_1, x_inst_2
#> Second stage: Dep. Var.: y
#> Observations: 150
#> Standard-errors: IID
#>              Estimate Std. Error  t value   Pr(>|t|)
#> (Intercept)  1.831380   0.411435  4.45121 1.6844e-05 ***
#> fit_x_endo_1 0.444982   0.022086 20.14744  < 2.2e-16 ***
#> fit_x_endo_2 0.639916   0.307376  2.08186 3.9100e-02 *
#> x1           0.565095   0.084715  6.67051 4.9180e-10 ***
```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> RMSE: 0.398842   Adj. R2: 0.761653
#> F-test (1st stage), x_endo_1: stat = 903.2    , p < 2.2e-16 , on 2 and 146 DoF.
#> F-test (1st stage), x_endo_2: stat =   3.25828, p = 0.041268, on 2 and 146 DoF.
#>                   Wu-Hausman: stat =   6.79183, p = 0.001518, on 2 and 144 DoF.

res_est_iv <- est_iv$coeftable |>
    rownames_to_column()


coef_of_interest <-
    res_est_iv[res_est_iv$rowname == "fit_x_endo_1", "Estimate"]
se_of_interest <-
    res_est_iv[res_est_iv$rowname == "fit_x_endo_1", "Std. Error"]
fstat_1st <- fitstat(est_iv, type = "ivf1")[[1]]$stat

# To get the correct SE based on 1st-stage F-stat (This result is similar without adju
# the results are the new CIS and p.value
tF(coef = coef_of_interest, se = se_of_interest, Fstat = fstat_1st) |>
    causalverse::nice_tab(5)
#>          F   cF    Coef      SE        t  CI2.5. CI97.5. p.value
#> 1 903.1628 1.96 0.44498 0.02209 20.14744 0.40169 0.48827       0

# We can try to see a different 1st-stage F-stat and how it changes the results
tF(coef = coef_of_interest, se = se_of_interest, Fstat = 2) |>
    causalverse::nice_tab(5)
#>   F    cF    Coef      SE        t  CI2.5. CI97.5. p.value
#> 1 2 18.66 0.44498 0.02209 20.14744 0.03285 0.85711 0.03432
```

33.3.3 AK approach

(Angrist and Kolesár, 2023)

33.4 Testing Assumptions

$$Y = \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

where

- X_1 are exogenous variables
- X_2 are endogenous variables
- Z are instrumental variables

If Z satisfies the relevance condition, it means $Cov(Z, X_2) \neq 0$

This is important because we need this to be able to estimate β_2 where

$$\beta_2 = \frac{Cov(Z, Y)}{Cov(Z, X_2)}$$

If Z satisfies the exogeneity condition, $E[Z\epsilon] = 0$, this can achieve by

- Z having no direct effect on Y except through X_2
- In the presence of omitted variable, Z is uncorrelated with this variable.

If we just want to know the effect of Z on Y (**reduced form**) where the coefficient of Z is

$$\rho = \frac{Cov(Y, Z)}{Var(Z)}$$

and this effect is only through X_2 (by the exclusion restriction assumption).

We can also consistently estimate the effect of Z on X (**first stage**) where the the coefficient of X_2 is

$$\pi = \frac{Cov(X_2, Z)}{Var(Z)}$$

and the IV estimate is

$$\beta_2 = \frac{Cov(Y, Z)}{Cov(X_2, Z)} = \frac{\rho}{\pi}$$

33.4.1 Relevance Assumption

- **Weak instruments**: can explain little variation in the endogenous regressor
  - Coefficient estimate of the endogenous variable will be inaccurate.
  - For cases where weak instruments are unavoidable, Moreira (2003) proposes the conditional likelihood ratio test for robust inference. This test is considered approximately optimal for weak instrument scenarios (Andrews et al., 2008; Andrews and Marmer, 2008).
- Rule of thumb:
  - Compute F-statistic in the first-stage, where it should be greater than 10. But this is discouraged now by Lee et al. (2022)
  - use `linearHypothesis()` to see only instrument coefficients.

First-Stage F-Test

In the context of a two-stage least squares (2SLS) setup where you are estimating the equation:

$$Y = X\beta + \epsilon$$

and X is endogenous, you typically estimate a first-stage regression of:

$$X = Z\pi + u$$

where Z is the instrument.

The first-stage F-test evaluates the joint significance of the instruments in this first stage:

$$F = \frac{(SSR_r - SSR_{ur})/q}{SSR_{ur}/(n-k-1)}$$

where:

- SSR_r is the sum of squared residuals from the restricted model (no instruments, just the constant).
- SSR_{ur} is the sum of squared residuals from the unrestricted model (with instruments).
- q is the number of instruments excluded from the main equation.
- n is the number of observations.
- k is the number of explanatory variables excluding the instruments.

Cragg-Donald Test

The Cragg-Donald statistic is essentially the same as the Wald statistic of the joint significance of the instruments in the first stage, and it's used specifically when you have multiple endogenous regressors. It's calculated as:

$$CD = n \times (R_{ur}^2 - R_r^2)$$

where:

- R_{ur}^2 and R_r^2 are the R-squared values from the unrestricted and restricted models respectively.
- n is the number of observations.

For one endogenous variable, the Cragg-Donald test results should align closely with those from Stock and Yogo. The Anderson canonical correlation test, a likelihood ratio test, also works under similar conditions, contrasting with Cragg-Donald's Wald statistic approach. Both are valid with one endogenous variable and at least one instrument.

Stock-Yogo Weak IV Test

The Stock-Yogo test does not directly compute a statistic like the F-test or Cragg-Donald, but rather uses pre-computed critical values to assess the strength of instruments. It often uses the eigenvalues derived from the concentration matrix:

$$S = \frac{1}{n}(Z'X)(X'Z)$$

where Z is the matrix of instruments and X is the matrix of endogenous regressors.

Stock and Yogo provide critical values for different scenarios (bias, size distortion) for a given number of instruments and endogenous regressors, based on the smallest eigenvalue of S. The test compares these eigenvalues against critical values that correspond to thresholds of permissible bias or size distortion in a 2SLS estimator.

- **Critical Values and Test Conditions**: The critical values derived by Stock and Yogo depend on the level of acceptable bias, the number of endogenous regressors, and the number of instruments. For example, with a 5% maximum acceptable bias, one endogenous variable, and three instruments, the critical value for a sufficient first stage F-statistic is 13.91. Note that this framework requires at least two overidentifying degree of freedom.

Comparison

Test	Description	Focus	Usage
First-Stage F-Test	Evaluates the joint significance of instruments in the first stage.	Predictive power of instruments for the endogenous variable.	Simplest and most direct test, widely used especially with a single endogenous variable. Rule of thumb: F $<$ 10 suggests weak instruments.
Cragg-Donald Test	Wald statistic for joint significance of instruments.	Joint strength of multiple instruments with multiple endogenous variables.	More appropriate in complex IV setups with multiple endogenous variables. Compares statistic against critical values for assessing instrument strength.

Test	Description	Focus	Usage
Stock-Yogo Weak IV Test	Compares test statistic to predetermined critical values.	Minimizing size distortions and bias from weak instruments.	Theoretical evaluation of instrument strength, ensuring the reliability of 2SLS estimates against specific thresholds of bias or size distortion.

All the mentioned tests (Stock Yogo, Cragg-Donald, Anderson canonical correlation test) assume errors are independently and identically distributed. If this assumption is violated, the Kleinbergen-Paap test is robust against violations of the iid assumption and can be applied even with a single endogenous variable and instrument, provided the model is properly identified (Baum and Schaffer, 2021).

33.4.1.1 Weak Instrument Tests

33.4.1.2 Cragg-Donald

(Cragg and Donald, 1993)

Similar to the first-stage F-statistic

```
library(cragg)
library(AER) # for dataaset
data("WeakInstrument")

cragg_donald(
    # control variables
    X = ~ 1,
    # endogeneous variables
    D = ~ x,
    # instrument variables
    Z = ~ z,
    data = WeakInstrument
)
#> Cragg-Donald test for weak instruments:
#>
#>      Data:                        WeakInstrument
#>      Controls:                    ~1
#>      Treatments:                  ~x
#>      Instruments:                 ~z
#>
#>      Cragg-Donald Statistic:        4.566136
#>      Df:                                 198
```

Large CD statistic implies that the instruments are strong, but not in our case

here. But to judge it against some critical value, we have to look at Stock-Yogo

33.4.1.3 Stock-Yogo

Stock and Yogo (2002) set the critical values such that the bias is less then 10% (default)

H_0 : Instruments are weak

H_1 : Instruments are not weak

```
library(cragg)
library(AER) # for dataaset
data("WeakInstrument")
stock_yogo_test(
    # control variables
    X = ~ 1,
    # endogeneous variables
    D = ~ x,
    # instrument variables
    Z = ~ z,
    size_bias = "bias",
    data = WeakInstrument
)
```

The CD statistic should be bigger than the set critical value to be considered strong instruments.

33.4.1.4 Anderson-Rubin

33.4.1.5 Stock-Wright

33.4.2 Exogeneity Assumption

The local average treatment effect (LATE) is defined as:

$$\text{LATE} = \frac{\text{reduced form}}{\text{first stage}} = \frac{\rho}{\phi}$$

This implies that the reduced form (ρ) is the product of the first stage (ϕ) and LATE:

$$\rho = \phi \times \text{LATE}$$

Thus, if the first stage (ϕ) is 0, the reduced form (ρ) should also be 0.

```
# Load necessary libraries
library(shiny)
library(AER)  # for ivreg
```

```r
library(ggplot2)  # for visualization
library(dplyr)  # for data manipulation

# Function to simulate the dataset
simulate_iv_data <- function(n, beta, phi, direct_effect) {
  Z <- rnorm(n)
  epsilon_x <- rnorm(n)
  epsilon_y <- rnorm(n)
  X <- phi * Z + epsilon_x
  Y <- beta * X + direct_effect * Z + epsilon_y
  data <- data.frame(Y = Y, X = X, Z = Z)
  return(data)
}

# Function to run the simulations and calculate the effects
run_simulation <- function(n, beta, phi, direct_effect) {
  # Simulate the data
  simulated_data <- simulate_iv_data(n, beta, phi, direct_effect)

  # Estimate first-stage effect (phi)
  first_stage <- lm(X ~ Z, data = simulated_data)
  phi <- coef(first_stage)["Z"]
  phi_ci <- confint(first_stage)["Z", ]

  # Estimate reduced-form effect (rho)
  reduced_form <- lm(Y ~ Z, data = simulated_data)
  rho <- coef(reduced_form)["Z"]
  rho_ci <- confint(reduced_form)["Z", ]

  # Estimate LATE using IV regression
  iv_model <- ivreg(Y ~ X | Z, data = simulated_data)
  iv_late <- coef(iv_model)["X"]
  iv_late_ci <- confint(iv_model)["X", ]

  # Calculate LATE as the ratio of reduced-form and first-stage coefficients
  calculated_late <- rho / phi
  calculated_late_se <- sqrt(
    (rho_ci[2] - rho)^2 / phi^2 + (rho * (phi_ci[2] - phi) / phi^2)^2
  )
  calculated_late_ci <- c(calculated_late - 1.96 * calculated_late_se,
                          calculated_late + 1.96 * calculated_late_se)

  # Return a list of results
  list(phi = phi,
       phi_ci = phi_ci,
```

```
        rho = rho,
        rho_ci = rho_ci,
        direct_effect = direct_effect,
        direct_effect_ci = c(direct_effect, direct_effect),  # Placeholder for direct effect CI
        iv_late = iv_late,
        iv_late_ci = iv_late_ci,
        calculated_late = calculated_late,
        calculated_late_ci = calculated_late_ci,
        true_effect = beta,
        true_effect_ci = c(beta, beta))  # Placeholder for true effect CI
}

# Define UI for the sliders
ui <- fluidPage(
  titlePanel("IV Model Simulation"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("beta", "True Effect of X on Y (beta):", min = 0, max = 1.0, value = 0.5, step
      sliderInput("phi", "First Stage Effect (phi):", min = 0, max = 1.0, value = 0.7, step = 0.1
      sliderInput("direct_effect", "Direct Effect of Z on Y:", min = -0.5, max = 0.5, value = 0,
    ),
    mainPanel(
      plotOutput("dotPlot")
    )
  )
)

# Define server logic to run the simulation and generate the plot
server <- function(input, output) {
  output$dotPlot <- renderPlot({
    # Run simulation
    results <- run_simulation(n = 1000, beta = input$beta, phi = input$phi, direct_effect = input

    # Prepare data for plotting
    plot_data <- data.frame(
      Effect = c("First Stage (phi)", "Reduced Form (rho)", "Direct Effect", "LATE (Ratio)", "LAT
      Value = c(results$phi, results$rho, results$direct_effect, results$calculated_late, results
      CI_Lower = c(results$phi_ci[1], results$rho_ci[1], results$direct_effect_ci[1], results$cal
      CI_Upper = c(results$phi_ci[2], results$rho_ci[2], results$direct_effect_ci[2], results$cal
    )

    # Create dot plot with confidence intervals
    ggplot(plot_data, aes(x = Effect, y = Value)) +
      geom_point(size = 3) +
      geom_errorbar(aes(ymin = CI_Lower, ymax = CI_Upper), width = 0.2) +
```

```
      labs(title = "IV Model Effects",
           y = "Coefficient Value") +
      coord_cartesian(ylim = c(-1, 1)) +  # Limits the y-axis to -1 to 1 but allows CI
      theme_minimal() +
      theme(axis.text.x = element_text(angle = 45, hjust = 1))
  })
}

# Run the application
shinyApp(ui = ui, server = server)
```

A statistically significant reduced form estimate without a corresponding first stage indicates an issue, suggesting an alternative channel linking instruments to outcomes or a direct effect of the IV on the outcome.

- **No Direct Effect**: When the direct effect is 0 and the first stage is 0, the reduced form is 0.
  - Note: Extremely rare cases with multiple additional paths that perfectly cancel each other out can also produce this result, but testing for all possible paths is impractical.
- **With Direct Effect**: When there is a direct effect of the IV on the outcome, the reduced form can be significantly different from 0, even if the first stage is 0.
  - This violates the exogeneity assumption, as the IV should only affect the outcome through the treatment variable.

To test the validity of the exogeneity assumption, we can use a sanity test:

- Identify groups for which the effects of instruments on the treatment variable are small and not significantly different from 0. The reduced form estimate for these groups should also be 0. These "no-first-stage samples" provide evidence of whether the exogeneity assumption is violated.

33.4.2.1 Overid Tests

- Wald test and Hausman test for exogeneity of X assuming Z is exogenous
  - People might prefer Wald test over Hausman test.
- Sargan (for 2SLS) is a simpler version of Hansen's J test (for IV-GMM)
- Modified J test (i.e., Regularized jacknife IV): can handle weak instruments and small sample size (Carrasco and Doukali, 2022) (also proposed a regularized F-test to test relevance assumption that is robust to heteroskedasticity).
- New advances: endogeneity robust inference in finite sample and sensitivity analysis of inference (Kiviet, 2020)

These tests that can provide evidence fo the validity of the over-identifying restrictions is not sufficient or necessary for the validity of the moment conditions (i.e., this assumption cannot be tested). (Deaton, 2010; Parente and Silva, 2012)

- The over-identifying restriction can still be valid even when the instruments are correlated with the error terms, but then in this case, what you're estimating is no longer your parameters of interest.
- Rejection of the over-identifying restrictions can also be the result of **parameter heterogeneity** (Angrist et al., 2000)

Why overid tests hold no value/info?

- Overidentifying restrictions are valid irrespective of the instruments' validity
  - Whenever instruments have the same motivation and are on the same scale, the estimated parameter of interests will be very close (Parente and Silva, 2012, p. 316)
- Overidentifying restriction are invalid when each instrument is valid
  - When the effect of your parameter of interest is heterogeneous (e.g., you have two groups with two different true effects), your first instrument can be correlated with your variable of interest only for the first group and your second interments can be correlated with your variable of interest only for the second group (i.e., each instrument is valid), and if you use each instrument, you can still identify the parameter of interest. However, if you use both of them, what you estimate is a mixture of the two groups. Hence, the overidentifying restriction will be invalid (because no single parameters can make the errors of the model orthogonal to both instruments). The result may seem confusing at first because if each subset of overidentifying restrictions is valid, the full set should also be valid. However, this interpretation is flawed because the residual's orthogonality to the instruments depends on the chosen set of instruments, and therefore the set of restrictions tested when using two sets of instruments together is not the same as the union of the sets of restrictions tested when using each set of instruments separately (Parente and Silva, 2012, p. 316)

These tests (of overidentifying restrictions) should be used to check whether different instruments identify the same parameters of interest, not to check their validity

(Hausman, 1983; Parente and Silva, 2012)

33.4.2.1.1 Wald Test Assuming that Z is exogenous (a valid instrument), we want to know whether X_2 is exogenous

1st stage:

$$X_2 = \hat{\alpha} Z + \hat{\epsilon}$$

2nd stage:

$$Y = \delta_0 X_1 + \delta_1 X_2 + \delta_2 \hat{\epsilon} + u$$

where

- $\hat{\epsilon}$ is the residuals from the 1st stage

The Wald test of exogeneity assumes

$$H_0 : \delta_2 = 0 H_1 : \delta_2 \neq 0$$

If you have more than one endogenous variable with more than one instrument, δ_2 is a vector of all residuals from all the first-stage equations. And the null hypothesis is that they are jointly equal 0.

If you reject this hypothesis, it means that X_2 is **not endogenous**. Hence, for this test, we do not want to reject the null hypothesis.

If the test is not sacrificially significant, we might just don't have enough information to reject the null.

When you have a valid instrument Z, whether X_2 is endogenous or exogenous, your coefficient estimates of X_2 should still be consistent. But if X_2 is exogenous, then 2SLS will be inefficient (i.e., larger standard errors).

Intuition:

$\hat{\epsilon}$ is the supposed endogenous part of X_2, When we regress Y on $\hat{\epsilon}$ and observe that its coefficient is not different from 0. It means that the exogenous part of X_2 can explain well the impact on Y, and there is no endogenous part.

33.4.2.1.2 Hausman's Test Similar to Wald Test and identical to Wald Test when we have homoskedasticity (i.e., homogeneity of variances). Because of this assumption, it's used less often than Wald Test

33.4.2.1.3 Hansen's J

- (Hansen, 1982)
- J-test (over-identifying restrictions test): test whether **additional** instruments are exogenous
  - Can only be applied in cases where you have more instruments than endogenous variables

    * $dim(Z) > dim(X_2)$
  - Assume at least one instrument within Z is exogenous

Procedure IV-GMM:

1. Obtain the residuals of the 2SLS estimation
2. Regress the residuals on all instruments and exogenous variables.
3. Test the joint hypothesis that all coefficients of the residuals across instruments are 0 (i.e., this is true when instruments are exogenous).
    1. Compute $J = mF$ where m is the number of instruments, and F is your equation F statistic (can you use `linearHypothesis()` again).
    2. If your exogeneity assumption is true, then $J \sim \chi^2_{m-k}$ where k is the number of endogenous variables.
4. If you reject this hypothesis, it can be that
    1. The first sets of instruments are invalid
    2. The second sets of instruments are invalid
    3. Both sets of instruments are invalid

Note: This test is only true when your residuals are homoskedastic.

For a heteroskedasticity-robust J-statistic, see (Carrasco and Doukali, 2022; Li et al., 2022)

33.4.2.1.4 Sargan Test (Sargan, 1958)

Similar to Hansen's J, but it assumes homoskedasticity

- Have to be careful when sample is not collected exogenously. As such, when you have choice-based sampling design, the sampling weights have to be considered to have consistent estimates. However, even if we apply sampling weights, the tests are not suitable because the iid assumption off errors are already violated. Hence, the test is invalid in this case (Pitt, 2011).
- If one has heteroskedasticity in its design, the Sargan test is invalid (Pitt, 2011)

33.5 Negative R^2

It's okay to have negative R^2 in the 2nd stage. We care more about consistent coefficient estimates.

R^2 has no statistical meaning in instrumental variable regression or 2 or 3SLS

$$R^2 = \frac{MSS}{TSS}$$

where

- MSS = model sum of squares (TSS- RSS)
- TSS = total sum of squares ($\sum(y-\bar{y})^2$)
- RSS = residual sum of squares ($\sum(y-Xb)^2$)

If $TSS > RSS$, then we have negative RSS and negative R^2. Since the predicted values of the endogenous variables are different from the endogenous variables themselves, the error that is used to calculate RSS can be different from the error in the second stage, and RSS in the second stage can be less than TSS.

33.6 Treatment Intensity

Two-Stage Least Squares (TSLS) can be used to estimate the average causal effect of variable treatment intensity, and it "identifies a weighted average of per-unit treatment effects along the length of a causal response function" (Angrist and Imbens, 1995, p. 431). For example

- Drug dosage
- Hours of exam prep on score (Powers and Swinton, 1984)
- Cigarette smoking on birth weights (Permutt and Hebel, 1989)
- Years of education
- Class size on test score (Angrist and Lavy, 1999)
- Sibship size on earning (Lavy et al., 2006)
- Social Media Adoption

The **average causal effect** here refers to the conditional expectation of the difference in outcomes between the treated and what would have happened in the counterfactual world.

Notes:

- We do not need a linearity assumption of the relationships between the dependent variable, treatment intensities, and instruments.

Example

In their original paper, Angrist and Imbens (1995) take the example of schooling effect on earnings where they have quarters of birth as the instrumental variable.

For each additional year of schooling, there can be an increase in earnings, and each additional year can be heterogeneous (both in the sense that grade 9th to grade 10th is qualitatively different and one can change to a different school).

$$Y = \gamma_0 + \gamma_1 X_1 + \rho S + \epsilon$$

where

- S is years of schooling (i.e., endogenous regressor)
- ρ is the return to a year of schooling
- X_1 is a matrix of exogenous covariates

Schooling can also be related to the exogenous variable X_1

$$S = \delta_0 + X_1\delta_1 + X_2\delta_2 + \eta$$

where

- X_2 is an exogenous instrument
- δ_2 is the coefficient of the instrument

by using only the fitted value in the second, the TSLS can give a consistent estimate of the effect of schooling on earning

$$Y = \gamma_0 + X_1\gamma - 1 + \rho\hat{S} + \nu$$

To give ρ a causal interpretation,

1. We first have to have the SUTVA (stable unit treatment value assumption), where the potential outcomes of the same person with different years of schooling are independent.
2. When ρ has a probability limit equal to a weighted average of $E[Y_j - Y_{j-1}]\forall j$

Even though the first bullet point is not trivial, most of the time we don't have to defend much about it in a research article, the second bullet point is the harder one to argue and only apply to certain cases.

33.7 Control Function

Also known as **two-stage residual inclusion**

Resources:

- Binary outcome and binary endogenous variable application (Tchetgen Tchetgen, 2014)
  - In rare events: we use a logistic model in the 2nd stage
  - In non-rare events: use risk ratio regression in the 2nd stage
- Application in marketing for consumer choice model (Petrin and Train, 2010)

Notes

- This approach is better suited for models with nonadditive errors (e.g., discrete choice models), or binary endogenous model, binary response variable, etc.

$$Y = g(X) + UX = \pi(Z) + VE(U|Z, V) = E(U|V)E(V|Z) = 0$$

Under control function approach,

$$E(Y|Z, V) = g(X) + E(U|Z, V) = g(X) + E(U|V) = g(X) + h(V)$$

where $h(V)$ is the control function that models the endogeneity

1. Linear in parameters

a. Linear Endogenous Variables:
   - The control function function approach is identical to the usual 2SLS estimator

b. Nonlinear Endogenous Variables:
   - The control function is different from the 2SLS estimator

c. Nonlinear in parameters:
   - The CF function is superior than the 2SLS estimator

33.7.1 Simulation

```
library(fixest)
library(tidyverse)
library(modelsummary)

# Set the seed for reproducibility
set.seed(123)
n = 10000
# Generate the exogenous variable from a normal distribution
exogenous <- rnorm(n, mean = 5, sd = 1)

# Generate the omitted variable as a function of the exogenous variable
omitted <- rnorm(n, mean = 2, sd = 1)

# Generate the endogenous variable as a function of the omitted variable and the exoge
endogenous <- 5 * omitted + 2 * exogenous + rnorm(n, mean = 0, sd = 1)

# nonlinear endogenous variable
endogenous_nonlinear <- 5 * omitted^2 + 2 * exogenous + rnorm(100, mean = 0, sd = 1)

unrelated <- rexp(n, rate = 1)
```

```r
# Generate the response variable as a function of the endogenous variable and the omitted variabl
response <- 4 +  3 * endogenous + 6 * omitted + rnorm(n, mean = 0, sd = 1)

response_nonlinear <- 4 +  3 * endogenous_nonlinear + 6 * omitted + rnorm(n, mean = 0, sd = 1)

response_nonlinear_para <- 4 +  3 * endogenous ^ 2 + 6 * omitted + rnorm(n, mean = 0, sd = 1)


# Combine the variables into a data frame
my_data <-
    data.frame(
        exogenous,
        omitted,
        endogenous,
        response,
        unrelated,
        response,
        response_nonlinear,
        response_nonlinear_para
    )

# View the first few rows of the data frame
# head(my_data)

wo_omitted <- feols(response ~ endogenous + sw0(unrelated), data = my_data)
w_omitted  <- feols(response ~ endogenous + omitted + unrelated, data = my_data)


# ivreg::ivreg(response ~ endogenous + unrelated | exogenous, data = my_data)
iv <- feols(response ~ 1 + sw0(unrelated) | endogenous ~ exogenous, data = my_data)

etable(
    wo_omitted,
    w_omitted,
    iv,
    digits = 2
    # vcov = list("each", "iid", "hetero")
)
#>                   wo_omitted.1   wo_omitted.2      w_omitted           iv.1
#> Dependent Var.:       response       response       response       response
#>
#> Constant        -3.9*** (0.10) -4.0*** (0.10)  4.0*** (0.05) 15.7*** (0.59)
#> endogenous      4.0*** (0.005) 4.0*** (0.005) 3.0*** (0.004)  3.0*** (0.03)
#> unrelated                          0.03 (0.03)  0.002 (0.010)
#> omitted                                         6.0*** (0.02)
```

```
#> _______________ ______________ ______________ ______________ ______________
#> S.E. type                   IID            IID            IID             IID
#> Observations             10,000         10,000         10,000         10,000
#> R2                      0.98566        0.98567        0.99803        0.92608
#> Adj. R2                 0.98566        0.98566        0.99803        0.92607
#>
#>                            iv.2
#> Dependent Var.:        response
#>
#> Constant         15.6*** (0.59)
#> endogenous        3.0*** (0.03)
#> unrelated          0.10. (0.06)
#> omitted
#> _______________ ______________
#> S.E. type                   IID
#> Observations             10,000
#> R2                      0.92610
#> Adj. R2                 0.92608
#> ---
#> Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Linear in parameter and linear in endogenous variable

```
# manual
# 2SLS
first_stage = lm(endogenous ~ exogenous, data = my_data)
new_data = cbind(my_data, new_endogenous = predict(first_stage, my_data))
second_stage = lm(response ~ new_endogenous, data = new_data)
summary(second_stage)
#>
#> Call:
#> lm(formula = response ~ new_endogenous, data = new_data)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -77.683 -14.374  -0.107  14.289  78.274
#>
#> Coefficients:
#>                Estimate Std. Error t value Pr(>|t|)
#> (Intercept)     15.6743     2.0819   7.529 5.57e-14 ***
#> new_endogenous   3.0142     0.1039  29.025  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 21.26 on 9998 degrees of freedom
#> Multiple R-squared:  0.07771,    Adjusted R-squared:  0.07762
```

```
#> F-statistic: 842.4 on 1 and 9998 DF,  p-value: < 2.2e-16

new_data_cf = cbind(my_data, residual = resid(first_stage))
second_stage_cf = lm(response ~ endogenous + residual, data = new_data_cf)
summary(second_stage_cf)
#>
#> Call:
#> lm(formula = response ~ endogenous + residual, data = new_data_cf)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -5.360 -1.016  0.003  1.023  5.201
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 15.674265   0.149350   105.0   <2e-16 ***
#> endogenous   3.014202   0.007450   404.6   <2e-16 ***
#> residual     1.140920   0.008027   142.1   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.525 on 9997 degrees of freedom
#> Multiple R-squared:  0.9953, Adjusted R-squared:  0.9953
#> F-statistic: 1.048e+06 on 2 and 9997 DF,  p-value: < 2.2e-16

modelsummary(list(second_stage, second_stage_cf))
```

Nonlinear in endogenous variable

```
# 2SLS
first_stage = lm(endogenous_nonlinear ~ exogenous, data = my_data)

new_data = cbind(my_data, new_endogenous_nonlinear = predict(first_stage, my_data))
second_stage = lm(response_nonlinear ~ new_endogenous_nonlinear, data = new_data)
summary(second_stage)
#>
#> Call:
#> lm(formula = response_nonlinear ~ new_endogenous_nonlinear, data = new_data)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -94.43 -52.10 -15.29  36.50 446.08
#>
#> Coefficients:
#>                          Estimate Std. Error t value Pr(>|t|)
#> (Intercept)               15.3390    11.8175   1.298    0.194
```

	(1)	(2)
(Intercept)	15.674	15.674
	(2.082)	(0.149)
new_endogenous	3.014	
	(0.104)	
endogenous		3.014
		(0.007)
residual		1.141
		(0.008)
Num.Obs.	10 000	10 000
R2	0.078	0.995
R2 Adj.	0.078	0.995
AIC	89 520.9	36 826.8
BIC	89 542.5	36 855.6
Log.Lik.	−44 757.438	−18 409.377
F	842.424	1 048 263.304
RMSE	21.26	1.53

```
#> new_endogenous_nonlinear   3.0174     0.3376   8.938   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 69.51 on 9998 degrees of freedom
#> Multiple R-squared:  0.007927,   Adjusted R-squared:  0.007828
#> F-statistic: 79.89 on 1 and 9998 DF,  p-value: < 2.2e-16

new_data_cf = cbind(my_data, residual = resid(first_stage))
second_stage_cf = lm(response_nonlinear ~ endogenous_nonlinear + residual, data = new_d
summary(second_stage_cf)
#>
#> Call:
#> lm(formula = response_nonlinear ~ endogenous_nonlinear + residual,
#>     data = new_data_cf)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -17.5437  -0.8348   0.4614   1.4424   4.8154
#>
#> Coefficients:
#>                      Estimate Std. Error t value Pr(>|t|)
#> (Intercept)          15.33904    0.38459   39.88   <2e-16 ***
#> endogenous_nonlinear  3.01737    0.01099  274.64   <2e-16 ***
#> residual              0.24919    0.01104   22.58   <2e-16 ***
```

	(1)	(2)
(Intercept)	15.339	15.339
	(11.817)	(0.385)
new_endogenous_nonlinear	3.017	
	(0.338)	
endogenous_nonlinear		3.017
		(0.011)
residual		0.249
		(0.011)
Num.Obs.	10 000	10 000
R2	0.008	0.999
R2 Adj.	0.008	0.999
AIC	113 211.6	44 709.6
BIC	113 233.2	44 738.4
Log.Lik.	−56 602.782	−22 350.801
F	79.887	4 752 573.052
RMSE	69.50	2.26

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 2.262 on 9997 degrees of freedom
#> Multiple R-squared:  0.9989, Adjusted R-squared:  0.9989
#> F-statistic: 4.753e+06 on 2 and 9997 DF,  p-value: < 2.2e-16

modelsummary(list(second_stage, second_stage_cf))
```

Nonlinear in parameters

```
# 2SLS
first_stage = lm(endogenous ~ exogenous, data = my_data)

new_data = cbind(my_data, new_endogenous = predict(first_stage, my_data))
second_stage = lm(response_nonlinear_para ~ new_endogenous, data = new_data)
summary(second_stage)
#>
#> Call:
#> lm(formula = response_nonlinear_para ~ new_endogenous, data = new_data)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -1536.5  -452.4   -80.7   368.4  3780.9
#>
#> Coefficients:
```

```
#>                  Estimate Std. Error t value Pr(>|t|)
#> (Intercept)     -1089.943     61.706  -17.66   <2e-16 ***
#> new_endogenous    119.829      3.078   38.93   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 630.2 on 9998 degrees of freedom
#> Multiple R-squared:  0.1316, Adjusted R-squared:  0.1316
#> F-statistic:  1516 on 1 and 9998 DF,  p-value: < 2.2e-16

new_data_cf = cbind(my_data, residual = resid(first_stage))
second_stage_cf = lm(response_nonlinear_para ~ endogenous_nonlinear + residual, data =
summary(second_stage_cf)
#>
#> Call:
#> lm(formula = response_nonlinear_para ~ endogenous_nonlinear +
#>     residual, data = new_data_cf)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -961.00 -139.32  -16.02  135.57 1403.62
#>
#> Coefficients:
#>                      Estimate Std. Error t value Pr(>|t|)
#> (Intercept)          678.1593     9.9177   68.38   <2e-16 ***
#> endogenous_nonlinear  17.7884     0.2759   64.46   <2e-16 ***
#> residual              52.5016     1.1552   45.45   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 231.9 on 9997 degrees of freedom
#> Multiple R-squared:  0.8824, Adjusted R-squared:  0.8824
#> F-statistic: 3.751e+04 on 2 and 9997 DF,  p-value: < 2.2e-16

modelsummary(list(second_stage, second_stage_cf))
```

33.8 New Advances

- Combine ML and IV (Singh et al., 2020)

	(1)	(2)
(Intercept)	−1089.943	678.159
	(61.706)	(9.918)
new_endogenous	119.829	
	(3.078)	
endogenous_nonlinear		17.788
		(0.276)
residual		52.502
		(1.155)
Num.Obs.	10 000	10 000
R2	0.132	0.882
R2 Adj.	0.132	0.882
AIC	157 302.4	137 311.3
BIC	157 324.1	137 340.1
Log.Lik.	−78 648.225	−68 651.628
F	1515.642	37 505.777
RMSE	630.10	231.88

Chapter 34

Matching Methods

Matching is a process that aims to close back doors - potential sources of bias - by constructing comparison groups that are similar according to a set of matching variables. This helps to ensure that any observed differences in outcomes between the treatment and comparison groups can be more confidently attributed to the treatment itself, rather than other factors that may differ between the groups.

Matching and DiD can use pre-treatment outcomes to correct for selection bias. From real world data and simulation, (Chabé-Ferret, 2015) found that matching generally underestimates the average causal effect and gets closer to the true effect with more number of pre-treatment outcomes. When selection bias is symmetric around the treatment date, DID is still consistent when implemented symmetrically (i.e., the same number of period before and after treatment). In cases where selection bias is asymmetric, the MC simulations show that Symmetric DID still performs better than Matching.

Matching is useful, but not a general solution to causal problems (Smith and Todd, 2005)

Assumption: Observables can identify the selection into the treatment and control groups

Identification: The exclusion restriction can be met conditional on the observables

Motivation

Effect of college quality on earnings

- They ultimately estimate the treatment effect on the treated of attending a top (high ACT) versus bottom (low ACT) quartile college

Example

Aaronson et al. (2007)

Do teachers qualifications (causally) affect student test scores?

Step 1:

$$Y_{ijt} = \delta_0 + Y_{ij(t-1)}\delta_1 + X_{it}\delta_2 + Z_{jt}\delta_3 + \epsilon_{ijt}$$

There can always be another variable

Any observable sorting is imperfect

Step 2:

$$Y_{ijst} = \alpha_0 + Y_{ij(t-1)}\alpha_1 + X_{it}\alpha_2 + Z_{jt}\alpha_3 + \gamma_s + u_{isjt}$$

- $\delta_3 > 0$
- $\delta_3 > \alpha_3$
- γ_s = school fixed effect

Sorting is less within school. Hence, we can introduce the school fixed effect

Step 3:

Find schools that look like they are putting students in class randomly (or as good as random) + we run step 2

$$\begin{aligned} Y_{isjt} = Y_{isj(t-1)}\lambda + X_{it}\alpha_1 + Z_{jt}\alpha_{21} \\ + (Z_{jt} \times D_i)\alpha_{22} + \gamma_5 + u_{isjt} \end{aligned}$$

- D_{it} is an element of X_{it}
- Z_{it} = teacher experience

$$D_{it} = \begin{cases} 1 & \text{if high poverty} \\ 0 & \text{otherwise} \end{cases}$$

$H_0 : \alpha_{22} = 0$ test for effect heterogeneity whether the effect of teacher experience (Z_{jt}) is different

- For low poverty is α_{21}
- For high poverty effect is $\alpha_{21} + \alpha_{22}$

Matching is **selection on observables** and only works if you have good observables.

Sufficient identification assumption under Selection on observable/ back-door criterion (based on Bernard Koch's presentation)

- Strong conditional ignorability
  - $Y(0), Y(1) \perp T|X$
  - No hidden confounders
- Overlap
  - $\forall x \in X, t \in \{0,1\} : p(T = t|X = x > 0$
  - All treatments have non-zero probability of being observed
- SUTVA/ Consistency
  - Treatment and outcomes of different subjects are independent

Relative to OLS

1. Matching makes the **common support** explicit (and changes default from "ignore" to "enforce")
2. Relaxes linear function form. Thus, less parametric.

It also helps if you have high ratio of controls to treatments.

For detail summary (Stuart, 2010)

Matching is defined as "any method that aims to equate (or"balance") the distribution of covariates in the treated and control groups." (Stuart, 2010, pp. 1)

Equivalently, matching is a selection on observables identifications strategy.

If you think your OLS estimate is biased, a matching estimate (almost surely) is too.

Unconditionally, consider

$$\begin{aligned}
&E(Y_i^T|T) - E(Y_i^C|C) + E(Y_i^C|T) - E(Y_i^C|T) \\
&\quad = E(Y_i^T - Y_i^C|T) + [E(Y_i^C|T) - E(Y_i^C|C)] \\
&\quad = E(Y_i^T - Y_i^C|T) + \text{selection bias}
\end{aligned}$$

where $E(Y_i^T - Y_i^C|T)$ is the causal inference that we want to know.

Randomization eliminates the selection bias.

If we don't have randomization, then $E(Y_i^C|T) \neq E(Y_i^C|C)$

Matching tries to do selection on observables $E(Y_i^C|X,T) = E(Y_i^C|X,C)$

Propensity Scores basically do $E(Y_i^C|P(X),T) = E(Y_i^C|P(X),C)$

Matching standard errors will exceed OLS standard errors

The treatment should have larger predictive power than the control because you use treatment to pick control (not control to pick treatment).

The average treatment effect (ATE) is

$$\frac{1}{N_T}\sum_{i=1}^{N_T}(Y_i^T - \frac{1}{N_{C_T}}\sum_{i=1}^{N_{C_T}} Y_i^C)$$

Since there is no closed-form solution for the standard error of the average treatment effect, we have to use bootstrapping to get standard error.

Professor Gary King advocates instead of using the word "matching", we should use "**pruning**" (i.e., deleting observations). It is a preprocessing step where it prunes nonmatches to make control variables less important in your analysis.

Without Matching

- **Imbalance data** leads to **model dependence** lead to a lot of **researcher discretion** leads to **bias**

With Matching

- We have balance data which essentially erase human discretion

Table 34.1: Table @ref(tab:Gary King - International Methods Colloquium talk 2015)

Balance Covariates	Complete Randomization	Fully Exact
Observed	On average	Exact
Unobserved	On average	On average

Fully blocked is superior on

- imbalance
- model dependence
- power
- efficiency
- bias
- research costs
- robustness

Matching is used when

- Outcomes are not available to select subjects for follow-up
- Outcomes are available to improve precision of the estimate (i.e., reduce bias)

Hence, we can only observe one outcome of a unit (either treated or control), we can think of this problem as missing data as well. Thus, this section is closely related to Imputation (Missing Data)

In observational studies, we cannot randomize the treatment effect. Subjects select their own treatments, which could introduce selection bias (i.e., systematic differences between group differences that confound the effects of response variable differences).

Matching is used to

- reduce model dependence
- diagnose balance in the dataset

Assumptions of matching:

1. treatment assignment is independent of potential outcomes given the covariates
   - $T \perp (Y(0), Y(1))|X$
   - known as ignorability, or ignorable, no hidden bias, or unconfounded.
   - You typically satisfy this assumption when unobserved covariates correlated with observed covariates.
     - But when unobserved covariates are unrelated to the observed covariates, you can use sensitivity analysis to check your result, or use "design sensitivity" (Heller et al., 2009)
2. positive probability of receiving treatment for all X
   - $0 < P(T = 1|X) < 1 \forall X$
3. Stable Unit Treatment value Assumption (SUTVA)
   - Outcomes of A are not affected by treatment of B.
     - Very hard in cases where there is "spillover" effects (interactions between control and treatment). To combat, we need to reduce interactions.

Generalization

- P_t: treated population -> N_t: random sample from treated
- P_c: control population -> N_c: random sample from control
- μ_i = means ; Σ_i = variance covariance matrix of the p covariates in group i ($i = t, c$)
- $X_j = p$ covariates of individual j
- T_j = treatment assignment
- Y_j = observed outcome

- Assume: $N_t < N_c$
- Treatment effect is $\tau(x) = R_1(x) - R_0(x)$ where
  - $R_1(x) = E(Y(1)|X)$
  - $R_0(x) = E(Y(0)|X)$
- Assume: parallel trends hence $\tau(x) = \tau \forall x$
  - If the parallel trends are not assumed, an average effect can be estimated.
- Common estimands:
  - Average effect of the treatment on the treated (**ATT**): effects on treatment group
  - Average treatment effect (**ATE**): effect on both treatment and control

Steps:

1. Define "closeness": decide distance measure to be used
   1. Which variables to include:
      1. Ignorability (no unobserved differences between treatment and control)
         1. Since cost of including unrelated variables is small, you should include as many as possible (unless sample size/power doesn't allow you to because of increased variance)
         2. Do not include variables that were affected by the treatment.
         3. Note: if a matching variable (i.e., heavy drug users) is highly correlated to the outcome variable (i.e., heavy drinkers) , you will be better to exclude it in the matching set.
   2. Which distance measures: more below
2. Matching methods
   1. Nearest neighbor matching
      1. Simple (greedy) matching: performs poorly when there is competition for controls.
      2. Optimal matching: considers global distance measure
      3. Ratio matching: to combat increase bias and reduced variation when you have k:1 matching, one can use approximations by Rubin and Thomas (1996).

4. With or without replacement: with replacement is typically better, but one needs to account for dependent in the matched sample when doing later analysis (can use frequency weights to combat).

2. Subclassification, Full Matching and Weighting

   Nearest neighbor matching assign is 0 (control) or 1 (treated), while these methods use weights between 0 and 1.

   1. Subclassification: distribution into multiple subclass (e.g., 5-10)
   2. Full matching: optimal ly minimize the average of the distances between each treated unit and each control unit within each matched set.
   3. Weighting adjustments: weighting technique uses propensity scores to estimate ATE. If the weights are extreme, the variance can be large not due to the underlying probabilities, but due to the estimation procure. To combat this, use (1) weight trimming, or (2) doubly -robust methods when propensity scores are used for weighing or matching.
      1. Inverse probability of treatment weighting (IPTW) $w_i = \frac{T_i}{\hat{e}_i} + \frac{1-T_i}{1-\hat{e}_i}$
      2. Odds $w_i = T_i + (1 - T_i)\frac{\hat{e}_i}{1-\hat{e}_i}$
      3. Kernel weighting (e.g., in economics) averages over multiple units in the control group.

3. Assessing Common Support

   - common support means overlapping of the propensity score distributions in the treatment and control groups. Propensity score is used to discard control units from the common support. Alternatively, convex hull of the covariates in the multi-dimensional space.

3. Assessing the quality of matched samples (Diagnose)

   - Balance = similarity of the empirical distribution of the full set of covariates in the matched treated and control groups. Equivalently, treatment is unrelated to the covariates
     - $\tilde{p}(X|T = 1) = \tilde{p}(X|T = 0)$ where $\tilde{p}$ is the empirical distribution.
   - Numerical Diagnostics
     1. standardized difference in means of each covariate (most common), also known as"standardized bias", "standardized difference in means".

2. standardized difference of means of the propensity score (should be < 0.25) (Rubin, 2001)
3. ratio of the variances of the propensity score in the treated and control groups (should be between 0.5 and 2). (Rubin, 2001)
4. For each covariate, the ratio fo the variance of the residuals orthogonal to the propensity score in the treated and control groups.

   Note: can't use hypothesis tests or p-values because of (1) in-sample property (not population), (2) conflation of changes in balance with changes in statistical power.

- Graphical Diagnostics
  - QQ plots
  - Empirical Distribution Plot

4. Estimate the treatment effect
   1. After k:1
      1. Need to account for weights when use matching with replacement.
   2. After Subclassification and Full Matching
      1. Weighting the subclass estimates by the number of treated units in each subclass for ATT
      2. Weighting by the overall number of individual in each subclass for ATE.
   3. Variance estimation: should incorporate uncertainties in both the matching procedure (step 3) and the estimation procedure (step 4)

Notes:

- With missing data, use generalized boosted models, or multiple imputation (Qu and Lipkovich, 2009)
- Violation of ignorable treatment assignment (i.e., unobservables affect treatment and outcome). control by
  - measure pre-treatment measure of the outcome variable
  - find the difference in outcomes between multiple control groups. If there is a significant difference, there is evidence for violation.
  - find the range of correlations between unobservables and both treatment assignment and outcome to nullify the significant effect.
- Choosing between methods

  - smallest standardized difference of mean across the largest number of covariates
  - minimize the standardized difference of means of a few particularly prognostic covariates
  - fest number of large standardized difference of means (> 0.25)
  - (Diamond and Sekhon, 2013) automates the process
- In practice
  - If ATE, ask if there is enough overlap of the treated and control groups' propensity score to estimate ATE, if not use ATT instead
  - If ATT, ask if there are controls across the full range of the treated group
- Choose matching method
  - If ATE, use IPTW or full matching
  - If ATT, and more controls than treated (at least 3 times), k:1 nearest neighbor without replacement
  - If ATT, and few controls , use subclassification, full matching, and weighting by the odds
- Diagnostic
  - If balance, use regression on matched samples
  - If imbalance on few covariates, treat them with Mahalanobis
  - If imbalance on many covariates, try k:1 matching with replacement

Ways to define the distance D_{ij}

1. Exact

$$D_{ij} = \begin{cases} 0, \text{ if } X_i = X_j, \\ \infty, \text{ if } X_i \neq X_j \end{cases}$$

An advanced is Coarsened Exact Matching

2. Mahalanobis

$$D_{ij} = (X_i - X_j)' \Sigma^{-1} (X_i - X_j)$$

where

Σ = variance covariance matrix of X in the

- control group if ATT is interested

- polled treatment and control groups if ATE is interested

3. Propensity score:

$$D_{ij} = |e_i - e_j|$$

where e_k = the propensity score for individual k

An advanced is Prognosis score (Hansen, 2008), but you have to know (i.e., specify) the relationship between the covariates and outcome.

4. Linear propensity score

$$D_{ij} = |logit(e_i) - logit(e_j)|$$

The exact and Mahalanobis are not good in high dimensional or non normally distributed X's cases.

We can combine Mahalanobis matching with propensity score calipers (Rubin and Thomas, 2000)

Other advanced methods for longitudinal settings

- marginal structural models (Robins et al., 2000)
- balanced risk set matching (Li et al., 2001)

Most matching methods are based on (ex-post)

- propensity score
- distance metric
- covariates

Packages

- `cem` Coarsened exact matching
- `Matching` Multivariate and propensity score matching with balance optimization
- `MatchIt` Nonparametric preprocessing for parametric causal inference. Have nearest neighbor, Mahalanobis, caliper, exact, full, optimal, subclassification
- `MatchingFrontier` optimize balance and sample size (King et al., 2017)
- `optmatch`optimal matching with variable ratio, optimal and full matching
- `PSAgraphics` Propensity score graphics
- `rbounds` sensitivity analysis with matched data, examine ignorable treatment assignment assumption

- `twang` weighting and analysis of non-equivalent groups
- `CBPS` covariate balancing propensity score. Can also be used in the longitudinal setting with marginal structural models.
- `PanelMatch` based on Imai, Kim, and Wang (2018)

Matching	Regression
Not as sensitive to the functional form of the covariates	can estimate the effect of a continuous treatment
Easier to asses whether it's working	estimate the effect of all the variables (not just the treatment)
Easier to explain	
allows a nice visualization of an evaluation	
If you treatment is fairly rare, you may have a lot of control observations that are obviously no comparable	can estimate interactions of treatment with covariates
Less parametric	More parametric
Enforces common support (i.e., space where treatment and control have the same characteristics)	

However, the problem of **omitted variables** (i.e., those that affect both the outcome and whether observation was treated) - unobserved confounders is still present in matching methods.

Difference between matching and regression following Pischke's lecture

Suppose we want to estimate the effect of treatment on the treated

$$\begin{aligned}\delta_{TOT} &= E[Y_{1i} - Y_{0i}|D_i = 1] \\ &= E\{E[Y_{1i}|X_i, D_i = 1] \\ &- E[Y_{0i}|X_i, D_i = 1]|D_i = 1\} \quad \text{law of itereated expectations}\end{aligned}$$

Under conditional independence

$$E[Y_{0i}|X_i, D_i = 0] = E[Y_{0i}|X_i, D_i = 1]$$

then

$$\begin{aligned}\delta_{TOT} &= E\{E[Y_{1i}|X_i, D_i = 1] - E[Y_{0i}|X_i, D_i = 0]|D_i = 1\} \\ &= E\{E[y_i|X_i, D_i = 1] - E[y_i|X_i, D_i = 0]|D_i = 1\} \\ &= E[\delta_X|D_i = 1]\end{aligned}$$

where δ_X is an X-specific difference in means at covariate value X_i

When X_i is discrete, the matching estimand is

$$\delta_M = \sum_x \delta_x P(X_i = x | D_i = 1)$$

where $P(X_i = x|D_i = 1)$ is the probability mass function for X_i given $D_i = 1$ According to Bayes rule,

$$P(X_i = x|D_i = 1) = \frac{P(D_i = 1|X_i = x) \times P(X_i = x)}{P(D_i = 1)}$$

hence,

$$\begin{aligned}\delta_M &= \frac{\sum_x \delta_x P(D_i = 1|X_i = x)P(X_i = x)}{\sum_x P(D_i = 1|X_i = x)P(X_i = x)} \\ &= \sum_x \delta_x \frac{P(D_i = 1|X_i = x)P(X_i = x)}{\sum_x P(D_i = 1|X_i = x)P(X_i = x)}\end{aligned}$$

On the other hand, suppose we have regression

$$y_i = \sum_x d_{ix}\beta_x + \delta_R D_i + \epsilon_i$$

where

- d_{ix} = dummy that indicates $X_i = x$
- β_x = regression-effect for $X_i = x$
- δ_R = regression estimand where

$$\begin{aligned}\delta_R &= \frac{\sum_x \delta_x [P(D_i = 1|X_i = x)(1 - P(D_i = 1|X_i = x))]P(X_i = x)}{\sum_x [P(D_i = 1|X_i = x)(1 - P(D_i = 1|X_i = x))]P(X_i = x)} \\ &= \sum_x \delta_x \frac{[P(D_i = 1|X_i = x)(1 - P(D_i = 1|X_i = x))]P(X_i = x)}{\sum_x [P(D_i = 1|X_i = x)(1 - P(D_i = 1|X_i = x))]P(X_i = x)}\end{aligned}$$

the difference between the regression and matching estimand is the weights they use to combine the covariate specific treatment effect δ_x

Type	uses weights which depend on	interpretation	makes sense because
Matching	$P(D_i = 1\|X_i = x)$ the fraction of treated observations in a covariate cell (i.e., or the mean of D_i)	This is larger in cells with many treated observations.	we want the effect of treatment on the treated
Regression	$P(D_i = 1\|X_i = x)(1 - P(D_i = 1\|X_i))$ the variance of D_i in the covariate cell	This weight is largest in cells where there are half treated and half untreated observations. (this is the reason why we want to treat our sample so it is balanced, before running regular regression model, as mentioned above).	these cells will produce the lowest variance estimates of δ_x. If all the δ_x are the same, the most efficient estimand uses the lowest variance cells most heavily.

The goal of matching is to produce covariate balance (i.e., distributions of covariates in treatment and control groups are approximately similar as they would be in a successful randomized experiment).

34.1 Selection on Observables

34.1.1 MatchIt

Procedure typically involves (proposed by Noah Freifer using `MatchIt`)

1. planning
2. matching
3. checking (balance)
4. estimating the treatment effect

```
library(MatchIt)
data("lalonde")
```

examine `treat` on `re78`

1. Planning

- select type of effect to be estimated (e.g., mediation effect, conditional effect, marginal effect)
- select the target population
- select variables to match/balance (Austin, 2011) (VanderWeele, 2019)

2. Check Initial Imbalance

```
# No matching; constructing a pre-match matchit object
m.out0 <- matchit(
    formula(treat ~ age + educ + race
            + married + nodegree + re74 + re75, env = lalonde),
    data = data.frame(lalonde),
    method = NULL,
    # assess balance before matching
    distance = "glm" # logistic regression
)

# Checking balance prior to matching
summary(m.out0)
```

3. Matching

```
# 1:1 NN PS matching w/o replacement
m.out1 <- matchit(treat ~ age + educ,
                  data = lalonde,
                  method = "nearest",
                  distance = "glm")
m.out1
#> A matchit object
#>  - method: 1:1 nearest neighbor matching without replacement
#>  - distance: Propensity score
#>              - estimated with logistic regression
#>  - number of obs.: 614 (original), 370 (matched)
#>  - target estimand: ATT
#>  - covariates: age, educ
```

4. Check balance

Sometimes you have to make trade-off between balance and sample size.

```
# Checking balance after NN matching
summary(m.out1, un = FALSE)
#>
#> Call:
#> matchit(formula = treat ~ age + educ, data = lalonde, method = "nearest",
#>     distance = "glm")
#>
#> Summary of Balance for Matched Data:
```

```
#>          Means Treated Means Control Std. Mean Diff. Var. Ratio eCDF Mean
#> distance        0.3080        0.3077          0.0094     0.9963    0.0033
#> age            25.8162       25.8649         -0.0068     1.0300    0.0050
#> educ           10.3459       10.2865          0.0296     0.5886    0.0253
#>          eCDF Max Std. Pair Dist.
#> distance   0.0432          0.0146
#> age        0.0162          0.0597
#> educ       0.1189          0.8146
#>
#> Sample Sizes:
#>           Control Treated
#> All           429     185
#> Matched       185     185
#> Unmatched     244       0
#> Discarded       0       0

# examine visually
plot(m.out1, type = "jitter", interactive = FALSE)
```

```
plot(
    m.out1,
    type = "qq",
    interactive = FALSE,
    which.xs = c("age")
)
```

Try Full Match (i.e., every treated matches with one control, and every control with one treated).

```
# Full matching on a probit PS
m.out2 <- matchit(treat ~ age + educ,
                  data = lalonde,
                  method = "full",
                  distance = "glm",
                  link = "probit")
m.out2
#> A matchit object
#>  - method: Optimal full matching
#>  - distance: Propensity score
#>              - estimated with probit regression
#>  - number of obs.: 614 (original), 614 (matched)
#>  - target estimand: ATT
#>  - covariates: age, educ
```

Checking balance again

```
# Checking balance after full matching
summary(m.out2, un = FALSE)
#>
#> Call:
#> matchit(formula = treat ~ age + educ, data = lalonde, method = "full",
#>     distance = "glm", link = "probit")
#>
#> Summary of Balance for Matched Data:
```

```
#>          Means Treated Means Control Std. Mean Diff. Var. Ratio eCDF Mean
#> distance        0.3082        0.3081          0.0023     0.9815    0.0028
#> age            25.8162       25.8035          0.0018     0.9825    0.0062
#> educ           10.3459       10.2315          0.0569     0.4390    0.0481
#>          eCDF Max Std. Pair Dist.
#> distance   0.0270          0.0382
#> age        0.0249          0.1110
#> educ       0.1300          0.9805
#>
#> Sample Sizes:
#>               Control Treated
#> All            429.       185
#> Matched (ESS)  145.23     185
#> Matched        429.       185
#> Unmatched        0.         0
#> Discarded        0.         0

plot(summary(m.out2))
```

Exact Matching

```
# Full matching on a probit PS
m.out3 <-
    matchit(
        treat ~ age + educ,
        data = lalonde,
        method = "exact"
```

```
    )
m.out3
#> A matchit object
#>  - method: Exact matching
#>  - number of obs.: 614 (original), 332 (matched)
#>  - target estimand: ATT
#>  - covariates: age, educ
```

Subclassfication

```
m.out4 <- matchit(
    treat ~ age + educ,
    data = lalonde,
    method = "subclass"
)
m.out4
#> A matchit object
#>  - method: Subclassification (6 subclasses)
#>  - distance: Propensity score
#>              - estimated with logistic regression
#>  - number of obs.: 614 (original), 614 (matched)
#>  - target estimand: ATT
#>  - covariates: age, educ

# Or you can use in conjunction with "nearest"
m.out4 <- matchit(
    treat ~ age + educ,
    data = lalonde,
    method = "nearest",
    option = "subclass"
)
m.out4
#> A matchit object
#>  - method: 1:1 nearest neighbor matching without replacement
#>  - distance: Propensity score
#>              - estimated with logistic regression
#>  - number of obs.: 614 (original), 370 (matched)
#>  - target estimand: ATT
#>  - covariates: age, educ
```

Optimal Matching

```
m.out5 <- matchit(
    treat ~ age + educ,
    data = lalonde,
    method = "optimal",
```

```
    ratio = 2
)
m.out5
#> A matchit object
#>  - method: 2:1 optimal pair matching
#>  - distance: Propensity score
#>              - estimated with logistic regression
#>  - number of obs.: 614 (original), 555 (matched)
#>  - target estimand: ATT
#>  - covariates: age, educ
```

Genetic Matching

```
m.out6 <- matchit(
    treat ~ age + educ,
    data = lalonde,
    method = "genetic"
)
m.out6
#> A matchit object
#>  - method: 1:1 genetic matching without replacement
#>  - distance: Propensity score
#>              - estimated with logistic regression
#>  - number of obs.: 614 (original), 370 (matched)
#>  - target estimand: ATT
#>  - covariates: age, educ
```

4. Estimating the Treatment Effect

```
# get matched data
m.data1 <- match.data(m.out1)

head(m.data1)
#>      treat age educ   race married nodegree re74 re75       re78  distance
#> NSW1     1  37   11  black       1        1    0    0  9930.0460 0.2536942
#> NSW2     1  22    9 hispan       0        1    0    0  3595.8940 0.3245468
#> NSW3     1  30   12  black       0        0    0    0 24909.4500 0.2881139
#> NSW4     1  27   11  black       0        1    0    0  7506.1460 0.3016672
#> NSW5     1  33    8  black       0        1    0    0   289.7899 0.2683025
#> NSW6     1  22    9  black       0        1    0    0  4056.4940 0.3245468
#>      weights subclass
#> NSW1       1        1
#> NSW2       1       98
#> NSW3       1      109
#> NSW4       1      120
#> NSW5       1      131
#> NSW6       1      142
```

```r
library("lmtest") #coeftest
library("sandwich") #vcovCL

# imbalance matched dataset
fit1 <- lm(re78 ~ treat + age + educ ,
           data = m.data1,
           weights = weights)

coeftest(fit1, vcov. = vcovCL, cluster = ~subclass)
#>
#> t test of coefficients:
#>
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  -174.902   2445.013 -0.0715 0.943012
#> treat       -1139.085    780.399 -1.4596 0.145253
#> age           153.133     55.317  2.7683 0.005922 **
#> educ          358.577    163.860  2.1883 0.029278 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`treat` coefficient = estimated ATT

```r
# balance matched dataset
m.data2 <- match.data(m.out2)

fit2 <- lm(re78 ~ treat + age + educ ,
           data = m.data2, weights = weights)

coeftest(fit2, vcov. = vcovCL, cluster = ~subclass)
#>
#> t test of coefficients:
#>
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 2151.952   3141.152  0.6851  0.49355
#> treat       -725.184    703.297 -1.0311  0.30289
#> age          120.260     53.933  2.2298  0.02612 *
#> educ         175.693    241.694  0.7269  0.46755
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

When reporting, remember to mention

1. the matching specification (method, and additional options)
2. the distance measure (e.g., propensity score)
3. other methods, and rationale for the final chosen method.
4. balance statistics of the matched dataset.

5. number of matched, unmatched, discarded
6. estimation method for treatment effect.

34.1.2 designmatch

This package includes

- `distmatch` optimal distance matching
- `bmatch` optimal bipartile matching
- `cardmatch` optimal cardinality matching
- `profmatch` optimal profile matching
- `nmatch` optimal nonbipartile matching

```
library(designmatch)
```

34.1.3 MatchingFrontier

As mentioned in `MatchIt`, you have to make trade-off (also known as bias-variance trade-off) between balance and sample size. An automated procedure to optimize this trade-off is implemented in `MatchingFrontier` (King et al., 2017), which solves this joint optimization problem.

Following `MatchingFrontier` guide

```
# library(devtools)
# install_github('ChristopherLucas/MatchingFrontier')
library(MatchingFrontier)
data("lalonde")
# choose var to match on
match.on <-
    colnames(lalonde)[!(colnames(lalonde) %in% c('re78', 'treat'))]
match.on

# Mahanlanobis frontier (default)
mahal.frontier <-
    makeFrontier(
        dataset = lalonde,
        treatment = "treat",
        match.on = match.on
    )
mahal.frontier

# L1 frontier
L1.frontier <-
    makeFrontier(
```

```
        dataset = lalonde,
        treatment = 'treat',
        match.on = match.on,
        QOI = 'SATT',
        metric = 'L1',
        ratio = 'fixed'
    )
L1.frontier

# estimate effects along the frontier

# Set base form
my.form <-
    as.formula(re78 ~ treat + age + black + education
               + hispanic + married + nodegree + re74 + re75)

# Estimate effects for the mahalanobis frontier
mahal.estimates <-
    estimateEffects(
        mahal.frontier,
        're78 ~ treat',
        mod.dependence.formula = my.form,
        continuous.vars = c('age', 'education', 're74', 're75'),
        prop.estimated = .1,
        means.as.cutpoints = TRUE
    )

# Estimate effects for the L1 frontier
L1.estimates <-
    estimateEffects(
        L1.frontier,
        're78 ~ treat',
        mod.dependence.formula = my.form,
        continuous.vars = c('age', 'education', 're74', 're75'),
        prop.estimated = .1,
        means.as.cutpoints = TRUE
    )

# Plot covariates means
# plotPrunedMeans()


# Plot estimates (deprecated)
# plotEstimates(
#     L1.estimates,
```

```
#     ylim = c(-10000, 3000),
#     cex.lab = 1.4,
#     cex.axis = 1.4,
#     panel.first = grid(NULL, NULL, lwd = 2,)
# )

# Plot estimates
plotMeans(L1.frontier)


# parallel plot
parallelPlot(
    L1.frontier,
    N = 400,
    variables = c('age', 're74', 're75', 'black'),
    treated.col = 'blue',
    control.col = 'gray'
)

# export matched dataset
# take 400 units
matched.data <- generateDataset(L1.frontier, N = 400)
```

34.1.4 Propensity Scores

Even though I mention the propensity scores matching method here, it is no longer recommended to use such method in research and publication (King and Nielsen, 2019) because it increases

- imbalance
- inefficiency
- model dependence: small changes in the model specification lead to big changes in model results
- bias

(Abadie and Imbens, 2016)note

- The initial estimation of the propensity score influences the large sample distribution of the estimators.
- Adjustments are made to the large sample variances of these estimators for both ATE and ATT.
  - The adjustment for the ATE estimator is either negative or zero, indicating greater efficiency when matching on an estimated propensity score versus the true score in large samples.

  - For the ATET estimator, the sign of the adjustment depends on the data generating process. Neglecting the estimation error in the propensity score can lead to inaccurate confidence intervals for the ATT estimator, making them either too large or too small.

PSM tries to accomplish complete randomization while other methods try to achieve fully blocked. Hence, you probably better off use any other methods.

Propensity is "the probability of receiving the treatment given the observed covariates." (Rosenbaum and Rubin, 1985)

Equivalently, it can to understood as the probability of being treated.

$$e_i(X_i) = P(T_i = 1|X_i)$$

Estimation using

- logistic regression
- Non parametric methods:
  - boosted CART
  - generalized boosted models (gbm)

Steps by Gary King's slides

- reduce k elements of X to scalar
- $\pi_i \equiv P(T_i = 1|X) = \frac{1}{1+e^{X_i\beta}}$
- Distance $(X_c, X_t) = |\pi_c - \pi_t|$
- match each treated unit to the nearest control unit
- control units: not reused; pruned if unused
- prune matches if distances > caliper

In the best case scenario, you randomly prune, which increases imbalance

Other methods dominate because they try to match exactly hence

- $X_c = X_t \rightarrow \pi_c = \pi_t$ (exact match leads to equal propensity scores) but
- $\pi_c = \pi_t \nrightarrow X_c = X_t$ (equal propensity scores do not necessarily lead to exact match)

Notes:

- Do not include/control for irrelevant covariates because it leads your PSM to be more random, hence more imbalance
- Do not include for (Bhattacharya and Vogt, 2007) instrumental variable in the predictor set of a propensity score matching estimator. More generally,

using variables that do not control for potential confounders, even if they are predictive of the treatment, can result in biased estimates

What you left with after pruning is more important than what you start with then throw out.

Diagnostics:

- balance of the covariates
- no need to concern about collinearity
- can't use c-stat or stepwise because those model fit stat do not apply

Application

- Finance:
  - Hirtle et al. (2020) examine the impact of bank supervision on risk, profitability, and growth, using a matched sample approach to show that increased supervisory attention leads to less risky loan portfolios and reduced volatility without compromising profitability or growth.

34.1.4.1 Look Ahead Propensity Score Matching

- (Bapna et al., 2018)

34.1.5 Mahalanobis Distance

Approximates fully blocked experiment

Distance $(X_c, X_t) = \sqrt{(X_c - X_t)' S^{-1} (X_c - X_t)}$

where S^{-1} standardize the distance

In application we use Euclidean distance.

Prune unused control units, and prune matches if distance > caliper

34.1.6 Coarsened Exact Matching

Steps from Gray King's slides International Methods Colloquium talk 2015

- Temporarily coarsen X
- Apply exact matching to the coarsened $X, C(X)$
  - sort observation into strata, each with unique values of $C(X)$
  - prune stratum with 0 treated or 0 control units
- Pass on original (uncoarsened) units except those pruned

Properties:

- Monotonic imbalance bounding (MIB) matching method

  - maximum imbalance between the treated and control chosen ex ante

- meets congruence principle
- robust to measurement error
- can be implemented with multiple imputation
- works well for multi-category treatments

Assumptions:

- Ignorability (i.e., no omitted variable bias)

More detail in (Iacus et al., 2012)

Example by package's authors

```
library(cem)
data(LeLonde)

Le <- data.frame(na.omit(LeLonde)) # remove missing data
# treated and control groups
tr <- which(Le$treated==1)
ct <- which(Le$treated==0)
ntr <- length(tr)
nct <- length(ct)

# unadjusted, biased difference in means
mean(Le$re78[tr]) - mean(Le$re78[ct])
#> [1] 759.0479

# pre-treatment covariates
vars <-
    c(
        "age",
        "education",
        "black",
        "married",
        "nodegree",
        "re74",
        "re75",
        "hispanic",
        "u74",
        "u75",
        "q1"
    )

# overall imbalance statistics
imbalance(group=Le$treated, data=Le[vars]) # L1 = 0.902
```

```
#>
#> Multivariate Imbalance Measure: L1=0.902
#> Percentage of local common support: LCS=5.8%
#>
#> Univariate Imbalance Measures:
#>
#>               statistic   type           L1 min 25%      50%       75%
#> age        -0.252373042 (diff) 5.102041e-03   0   0   0.0000   -1.0000
#> education   0.153634710 (diff) 8.463851e-02   1   0   1.0000    1.0000
#> black      -0.010322734 (diff) 1.032273e-02   0   0   0.0000    0.0000
#> married    -0.009551495 (diff) 9.551495e-03   0   0   0.0000    0.0000
#> nodegree   -0.081217371 (diff) 8.121737e-02   0  -1   0.0000    0.0000
#> re74      -18.160446880 (diff) 5.551115e-17   0   0 284.0715  806.3452
#> re75      101.501761679 (diff) 5.551115e-17   0   0 485.6310 1238.4114
#> hispanic   -0.010144756 (diff) 1.014476e-02   0   0   0.0000    0.0000
#> u74        -0.045582186 (diff) 4.558219e-02   0   0   0.0000    0.0000
#> u75        -0.065555292 (diff) 6.555529e-02   0   0   0.0000    0.0000
#> q1          7.494021189 (Chi2) 1.067078e-01  NA  NA       NA        NA
#>                  max
#> age           -6.0000
#> education      1.0000
#> black          0.0000
#> married        0.0000
#> nodegree       0.0000
#> re74       -2139.0195
#> re75         490.3945
#> hispanic       0.0000
#> u74            0.0000
#> u75            0.0000
#> q1                 NA

# drop other variables that are not pre - treatmentt matching variables
todrop <- c("treated", "re78")
imbalance(group=Le$treated, data=Le, drop=todrop)
#>
#> Multivariate Imbalance Measure: L1=0.902
#> Percentage of local common support: LCS=5.8%
#>
#> Univariate Imbalance Measures:
#>
#>               statistic   type           L1 min 25%      50%       75%
#> age        -0.252373042 (diff) 5.102041e-03   0   0   0.0000   -1.0000
#> education   0.153634710 (diff) 8.463851e-02   1   0   1.0000    1.0000
#> black      -0.010322734 (diff) 1.032273e-02   0   0   0.0000    0.0000
#> married    -0.009551495 (diff) 9.551495e-03   0   0   0.0000    0.0000
```

```
#> nodegree   -0.081217371 (diff) 8.121737e-02   0  -1   0.0000    0.0000
#> re74      -18.160446880 (diff) 5.551115e-17   0   0 284.0715  806.3452
#> re75      101.501761679 (diff) 5.551115e-17   0   0 485.6310 1238.4114
#> hispanic   -0.010144756 (diff) 1.014476e-02   0   0   0.0000    0.0000
#> u74        -0.045582186 (diff) 4.558219e-02   0   0   0.0000    0.0000
#> u75        -0.065555292 (diff) 6.555529e-02   0   0   0.0000    0.0000
#> q1          7.494021189 (Chi2) 1.067078e-01  NA  NA       NA        NA
#>                   max
#> age           -6.0000
#> education      1.0000
#> black          0.0000
#> married        0.0000
#> nodegree       0.0000
#> re74       -2139.0195
#> re75         490.3945
#> hispanic       0.0000
#> u74            0.0000
#> u75            0.0000
#> q1                 NA
```

automated coarsening

```
mat <-
    cem(
        treatment = "treated",
        data = Le,
        drop = "re78",
        keep.all = TRUE
    )
#>
#> Using 'treated'='1' as baseline group
mat
#>            G0  G1
#> All       392 258
#> Matched    95  84
#> Unmatched 297 174

# mat$w
```

coarsening by explicit user choice

```
# categorial variables
levels(Le$q1) # grouping option
#> [1] "agree"             "disagree"          "neutral"
#> [4] "no opinion"        "strongly agree"    "strongly disagree"
q1.grp <-
    list(
```

```
        c("strongly agree", "agree"),
        c("neutral", "no opinion"),
        c("strongly disagree", "disagree")
    ) # if you want ordered categories

# continuous variables
table(Le$education)
#>
#>   3   4   5   6   7   8   9  10  11  12  13  14  15
#>   1   5   4   6  12  55 106 146 173 113  19   9   1
educut <- c(0, 6.5, 8.5, 12.5, 17)  # use cutpoints

mat1 <-
    cem(
        treatment = "treated",
        data = Le,
        drop = "re78",
        cutpoints = list(education = educut),
        grouping = list(q1 = q1.grp)
    )
#>
#> Using 'treated'='1' as baseline group
mat1
#>            G0  G1
#> All       392 258
#> Matched   158 115
#> Unmatched 234 143
```

- Can also use progressive coarsening method to control the number of matches.
- `cem` can also handle some missingness.

34.1.7 Genetic Matching

- GM uses iterative checking process of propensity scores, which combines propensity scores and Mahalanobis distance.
  - GenMatch (Diamond and Sekhon, 2013)
- GM is arguably "superior" method than nearest neighbor or full matching in imbalanced data
- Use a genetic search algorithm to find weights for each covariate such that we have optimal balance.
- Implementation

  - could use *with replacement*
  - balance can be based on
    * paired t-tests (dichotomous variables)
    * Kolmogorov-Smirnov (multinomial and continuous)

Packages

`Matching`

```
library(Matching)
data(lalonde)
attach(lalonde)

#The covariates we want to match on
X = cbind(age, educ, black, hisp, married, nodegr, u74, u75, re75, re74)

#The covariates we want to obtain balance on
BalanceMat <-
    cbind(age,
          educ,
          black,
          hisp,
          married,
          nodegr,
          u74,
          u75,
          re75,
          re74,
          I(re74 * re75))

#
#Let's call GenMatch() to find the optimal weight to give each
#covariate in 'X' so as we have achieved balance on the covariates in
#'BalanceMat'. This is only an example so we want GenMatch to be quick
#so the population size has been set to be only 16 via the 'pop.size'
#option. This is *WAY* too small for actual problems.
#For details see http://sekhon.berkeley.edu/papers/MatchingJSS.pdf.
#
genout <-
    GenMatch(
        Tr = treat,
        X = X,
        BalanceMatrix = BalanceMat,
        estimand = "ATE",
        M = 1,
```

```
        pop.size = 16,
        max.generations = 10,
        wait.generations = 1
    )

#The outcome variable
Y=re78/1000

#
# Now that GenMatch() has found the optimal weights, let's estimate
# our causal effect of interest using those weights
#
mout <-
    Match(
        Y = Y,
        Tr = treat,
        X = X,
        estimand = "ATE",
        Weight.matrix = genout
    )
summary(mout)

#
#Let's determine if balance has actually been obtained on the variables of interest
#
mb <-
    MatchBalance(
        treat ~ age + educ + black + hisp + married + nodegr
        + u74 + u75 + re75 + re74 + I(re74 * re75),
        match.out = mout,
        nboots = 500
    )
```

34.1.8 Entropy Balancing

(Hainmueller, 2012)

- Entropy balancing is a method for achieving covariate balance in observational studies with binary treatments.
- It uses a maximum entropy reweighting scheme to ensure that treatment and control groups are balanced based on sample moments.
- This method adjusts for inequalities in the covariate distributions, reducing dependence on the model used for estimating treatment effects.
- Entropy balancing improves balance across all included covariate moments

and removes the need for repetitive balance checking and iterative model searching.

34.1.9 Matching for high-dimensional data

One could reduce the number of dimensions using methods such as:

- Lasso (Gordon et al., 2019)
- Penalized logistic regression (Eckles and Bakshy, 2021)
- PCA (Principal Component Analysis)
- Locality Preserving Projections (LPP) (Li et al., 2016)
- Random projection
- Autoencoders (Ramachandra, 2018)

Additionally, one could jointly does dimension reduction while balancing the distributions of the control and treated groups (Yao et al., 2018).

34.1.10 Matching for time series-cross-section data

Examples: (Scheve and Stasavage, 2012) and (Acemoglu et al., 2019)

Identification strategy:

- Within-unit over-time variation
- within-time across-units variation

See DID with in and out treatment condition for details of this method

34.1.11 Matching for multiple treatments

In cases where you have multiple treatment groups, and you want to do matching, it's important to have the same baseline (control) group. For more details, see

- (McCaffrey et al., 2013)
- (Lopez and Gutman, 2017)
- (Zhao et al., 2021): also for continuous treatment

If you insist on using the `MatchIt` package, then see this answer

34.1.12 Matching for multi-level treatments

See (Yang et al., 2016)

Package in R `shuyang1987/multilevelMatching` on Github

34.1.13 Matching for repeated treatments

https://cran.r-project.org/web/packages/twang/vignettes/iptw.pdf

package in R `twang`

34.2 Selection on Unobservables

There are several ways one can deal with selection on unobservables:

1. Rosenbaum Bounds
2. Endogenous Sample Selection (i.e., Heckman-style correction): examine the λ term to see whether it's significant (sign of endogenous selection)
3. Relative Correlation Restrictions
4. Coefficient-stability Bounds

34.2.1 Rosenbaum Bounds

Examples in marketing

- (Oestreicher-Singer and Zalmanson, 2013): A range of 1.5 to 1.8 is important for the effect of the level of community participation of users on their willingness to pay for premium services.
- (Sun and Zhu, 2013): A factor of 1.5 is essential for understanding the relationship between the launch of an ad revenue-sharing program and the popularity of content.
- (Manchanda et al., 2015): A factor of 1.6 is required for the social dollar effect to be nullified.
- (Sudhir and Talukdar, 2015): A factor of 1.9 is needed for IT adoption to impact labor productivity, and 2.2 for IT adoption to affect floor productivity.
- (Proserpio and Zervas, 2017a): A factor of 2 is necessary for the firm's use of management responses to influence online reputation.
- (Zhang et al., 2022): A factor of 1.55 is critical for the acquisition of verified images to drive demand for Airbnb properties.
- (Chae et al., 2023): A factor of 27 (not a typo) is significant in how paywall suspensions affect subsequent subscription decisions.

General

- Matching Methods are favored for estimating treatment effects in observational data, offering advantages over regression methods because
  - It reduces reliance on functional form assumptions.

  - Assumes all selection-influencing covariates are observable; estimates are unbiased if no unobserved confounders are missed.

- Concerns arise when potentially relevant covariates are unmeasured.
  - Rosenbaum Bounds assess the overall sensitivity of coefficient estimates to hidden bias (Rosenbaum and Rosenbaum, 2002) without having knowledge (e.g., direction) of the bias. Because the unboservables that cause hidden bias have to both affect selection into treatment by a factor of Γ and predictive of outcome, this method is also known as **worst case analyses** (DiPrete and Gangl, 2004).

- Can't provide precise bounds on estimates of treatment effects (see Relative Correlation Restrictions)

- Typically, we show both p-value and H-L point estimate for each level of gamma Γ

With random treatment assignment, we can use the non-parametric test (Wilcoxon signed rank test) to see if there is treatment effect.

Without random treatment assignment (i.e., observational data), we cannot use this test. With Selection on Observables, we can use this test if we believe there are no unmeasured confounders. And this is where Rosenbaum (2002) can come in to talk about the believability of this notion.

In layman's terms, consider that the treatment assignment is based on a method where the odds of treatment for a unit and its control differ by a multiplier Γ

- For example, $\Gamma = 1$ means that the odds of assignment are identical, indicating random treatment assignment.
- Another example, $\Gamma = 2$, in the same matched pair, one unit is twice as likely to receive the treatment (due to unobservables).
- Since we can't know Γ with certainty, we run sensitivity analysis to see if the results change with different values of Γ
- This bias is the product of an unobservable that influences both treatment selection and outcome by a factor Γ (omitted variable bias)

In technical terms,

- **Treatment Assignment and Probability**:
  - Consider unit j with a probability π_j of receiving the treatment, and unit i with π_i.
  - Ideally, after matching, if there's no hidden bias, we'd have $\pi_i = \pi_j$.
  - However, observing $\pi_i \neq \pi_j$ raises questions about potential biases affecting our inference. This is evaluated using the odds ratio.
- **Odds Ratio and Hidden Bias**:
  - The odds of treatment for a unit j is defined as $\frac{\pi_j}{1-\pi_j}$.
  - The odds ratio between two matched units i and j is constrained by $\frac{1}{\Gamma} \leq \frac{\pi_i/(1-\pi_i)}{\pi_j/(1-\pi_j)} \leq \Gamma$.

    * If $\Gamma = 1$, it implies an absence of hidden bias.
    * If $\Gamma = 2$, the odds of receiving treatment could differ by up to a factor of 2 between the two units.
- **Sensitivity Analysis Using Gamma**:
  - The value of Γ helps measure the potential departure from a bias-free study.
  - Sensitivity analysis involves varying Γ to examine how inferences might change with the presence of hidden biases.
- **Incorporating Unobserved Covariates**:
  - Consider a scenario where unit i has observed covariates x_i and an unobserved covariate u_i, that both affect the outcome.
  - A logistic regression model could link the odds of assignment to these covariates: $\log(\frac{\pi_i}{1-\pi_i}) = \kappa x_i + \gamma u_i$, where γ represents the impact of the unobserved covariate.
- **Steps for Sensitivity Analysis** (We could create a table of different levels of Γ to assess how the magnitude of biases can affect our evidence of the treatment effect (estimate):
  1. Select a range of values for Γ (e.g., $1 \to 2$).
  2. Assess how the p-value or the magnitude of the treatment effect (Hodges Jr and Lehmann, 2011) (for more details, see (Hollander et al., 2013)) changes with varying Γ values.
  3. Employ specific randomization tests based on the type of outcome to establish bounds on inferences.
     - report the minimum value of Γ at which the treatment treat is nullified (i.e., become insignificant). And the literature's rules of thumb is that if $\Gamma > 2$, then we have strong evidence for our treatment effect is robust to large biases (Proserpio and Zervas, 2017b)

Notes:

- If we have treatment assignment is clustered (e.g., within school, within state) we need to adjust the bounds for clustered treatment assignment (Hansen et al., 2014) (similar to clustered standard errors).

Packages

- `rbounds` (Keele, 2010)
- `sensitivitymv` (Rosenbaum, 2015)

Since we typically assess our estimate sensitivity to unboservables after matching, we first do some matching.

```
library(MatchIt)
library(Matching)
data("lalonde")

matched <- MatchIt::matchit(
```

```
    treat ~ age + educ,
    data = lalonde,
    method = "nearest"
)
summary(matched)
#>
#> Call:
#> MatchIt::matchit(formula = treat ~ age + educ, data = lalonde,
#>     method = "nearest")
#>
#> Summary of Balance for All Data:
#>          Means Treated Means Control Std. Mean Diff. Var. Ratio eCDF Mean
#> distance        0.4203        0.4125          0.1689     1.2900    0.0431
#> age            25.8162       25.0538          0.1066     1.0278    0.0254
#> educ           10.3459       10.0885          0.1281     1.5513    0.0287
#>          eCDF Max
#> distance   0.1251
#> age        0.0652
#> educ       0.1265
#>
#> Summary of Balance for Matched Data:
#>          Means Treated Means Control Std. Mean Diff. Var. Ratio eCDF Mean
#> distance        0.4203        0.4179          0.0520     1.1691    0.0105
#> age            25.8162       25.5081          0.0431     1.1518    0.0148
#> educ           10.3459       10.2811          0.0323     1.5138    0.0224
#>          eCDF Max Std. Pair Dist.
#> distance   0.0595          0.0598
#> age        0.0486          0.5628
#> educ       0.0757          0.3602
#>
#> Sample Sizes:
#>           Control Treated
#> All           260     185
#> Matched       185     185
#> Unmatched      75       0
#> Discarded       0       0
matched_data <- match.data(matched)

treatment_group <- subset(matched_data, treat == 1)
control_group <- subset(matched_data, treat == 0)


library(rbounds)

# p-value sensitivity
```

```r
psens_res <-
    psens(treatment_group$re78,
          control_group$re78,
          Gamma = 2,
          GammaInc = .1)

psens_res
#>
#>  Rosenbaum Sensitivity Test for Wilcoxon Signed Rank P-Value
#>
#> Unconfounded estimate ....  0.0058
#>
#>  Gamma Lower bound Upper bound
#>    1.0      0.0058      0.0058
#>    1.1      0.0011      0.0235
#>    1.2      0.0002      0.0668
#>    1.3      0.0000      0.1458
#>    1.4      0.0000      0.2599
#>    1.5      0.0000      0.3967
#>    1.6      0.0000      0.5378
#>    1.7      0.0000      0.6664
#>    1.8      0.0000      0.7723
#>    1.9      0.0000      0.8523
#>    2.0      0.0000      0.9085
#>
#>  Note: Gamma is Odds of Differential Assignment To
#>  Treatment Due to Unobserved Factors
#>

# Hodges-Lehmann point estimate sensitivity
# median difference between treatment and control
hlsens_res <-
    hlsens(treatment_group$re78,
           control_group$re78,
           Gamma = 2,
           GammaInc = .1)
hlsens_res
#>
#>  Rosenbaum Sensitivity Test for Hodges-Lehmann Point Estimate
#>
#> Unconfounded estimate ....  1745.843
#>
#>  Gamma Lower bound Upper bound
#>    1.0 1745.800000      1745.8
#>    1.1 1139.100000      1865.6
```

```
#>    1.2  830.840000      2160.9
#>    1.3  533.740000      2462.4
#>    1.4  259.940000      2793.8
#>    1.5   -0.056912      3059.3
#>    1.6 -144.960000      3297.8
#>    1.7 -380.560000      3535.7
#>    1.8 -554.360000      3751.0
#>    1.9 -716.360000      4012.1
#>    2.0 -918.760000      4224.3
#>
#>  Note: Gamma is Odds of Differential Assignment To
#>  Treatment Due to Unobserved Factors
#>
```

For multiple control group matching

```
library(Matching)
library(MatchIt)

n_ratio <- 2
matched <- MatchIt::matchit(treat ~ age + educ ,
                    method = "nearest", ratio = n_ratio)
summary(matched)
matched_data <- match.data(matched)

mcontrol_res <- rbounds::mcontrol(
    y          = matched_data$re78,
    grp.id     = matched_data$subclass,
    treat.id   = matched_data$treat,
    group.size = n_ratio + 1,
    Gamma      = 2.5,
    GammaInc   = .1
)

mcontrol_res
```

`sensitivitymw` is faster than `sensitivitymw`. But `sensitivitymw` can match where matched sets can have differing numbers of controls (Rosenbaum, 2015).

```
library(sensitivitymv)
data(lead150)
head(lead150)
#>      [,1] [,2] [,3] [,4] [,5] [,6]
#> [1,] 1.40 1.23 2.24 0.96 1.90 1.14
#> [2,] 0.63 0.99 0.87 1.90 0.67 1.40
#> [3,] 1.98 0.82 0.66 0.58 1.00 1.30
#> [4,] 1.45 0.53 1.43 1.70 0.85 1.50
```

```
#> [5,] 1.60 1.70 0.63 1.05 1.08 0.92
#> [6,] 1.13 0.31 0.71 1.10 0.86 1.14
senmv(lead150,gamma=2,trim=2)
#> $pval
#> [1] 0.02665519
#>
#> $deviate
#> [1] 1.932398
#>
#> $statistic
#> [1] 27.97564
#>
#> $expectation
#> [1] 18.0064
#>
#> $variance
#> [1] 26.61524

library(sensitivitymw)
senmw(lead150,gamma=2,trim=2)
#> $pval
#> [1] 0.02665519
#>
#> $deviate
#> [1] 1.932398
#>
#> $statistic
#> [1] 27.97564
#>
#> $expectation
#> [1] 18.0064
#>
#> $variance
#> [1] 26.61524
```

34.2.2 Relative Correlation Restrictions

Examples in marketing

- (Manchanda et al., 2015): 3.23 for social dollar effect to be nullified
- (Chae et al., 2023): 6.69 (i.e., how much stronger the selection on unobservables has to be compared to the selection on observables to negate the result) for paywall suspensions affect subsequent subscription decisions
- (Sun and Zhu, 2013)

General

- Proposed by Altonji et al. (2005)
- Generalized by Krauth (2016)
- Estimate bounds of the treatment effects due to unobserved selection.

$$Y_i = X_i\beta + C_i\gamma + \epsilon_i$$

where

- β is the effect of interest
- C_i is the control variable
- Using OLS, $cor(X_i, \epsilon_i) = 0$

Under RCR analysis, we assume

$$cor(X_i, \epsilon_i) = \lambda cor(X_i, C_i\gamma)$$

where $\lambda \in (\lambda_l, \lambda_h)$

Choice of λ

- Strong assumption of no omitted variable bias (small
- If $\lambda = 0$, then $cor(X_i, \epsilon_i) = 0$
- If $\lambda = 1$, then $cor(X_i, \epsilon_i) = cor(X_i, C_i\gamma)$
- We typically examine $\lambda \in (0, 1)$

```
# remotes::install_github("bvkrauth/rcr/r/rcrbounds")
library(rcrbounds)
# rcrbounds::install_rcrpy()
data("ChickWeight")

rcr_res <-
    rcrbounds::rcr(weight ~ Time |
                       Diet, ChickWeight, rc_range = c(0, 10))
rcr_res
#>
#> Call:
#> rcrbounds::rcr(formula = weight ~ Time | Diet, data = ChickWeight,
#>     rc_range = c(0, 10))
#>
#> Coefficients:
#>     rcInf effectInf       rc0   effectL   effectH
#> 34.676505 71.989336 34.741955  7.447713  8.750492
```

```r
summary(rcr_res)
#>
#> Call:
#> rcrbounds::rcr(formula = weight ~ Time | Diet, data = ChickWeight,
#>     rc_range = c(0, 10))
#>
#> Coefficients:
#>            Estimate  Std. Error    t value      Pr(>|t|)
#> rcInf     34.676505  50.1295005  0.6917385  4.891016e-01
#> effectInf 71.989336 112.5711682  0.6395007  5.224973e-01
#> rc0       34.741955  58.7169195  0.5916856  5.540611e-01
#> effectL    7.447713   2.4276246  3.0679014  2.155677e-03
#> effectH    8.750492   0.2607671 33.5567355 7.180405e-247
#> ---
#> conservative confidence interval:
#>          2.5  %  97.5  %
#> effect 2.689656 9.261586

# hypothesis test for the coefficient
rcrbounds::effect_test(rcr_res, h0 = 0)
#> [1] 0.001234233
plot(rcr_res)
```

34.2.3 Coefficient-stability Bounds

- Developed by Oster (2019)

- Assess robustness to omitted variable bias by observing:
  1. Changes in the coefficient of interest
  2. Shifts in model R^2
- Refer Masten and Poirier (2022) for reverse sign problem.

Chapter 35

Interrupted Time Series

- Regression Discontinuity in Time
- Control for
  - Seasonable trends
  - Concurrent events
- Pros (Penfold and Zhang, 2013)
  - control for long-term trends
- Cons
  - Min of 8 data points before and 8 after an intervention
  - Multiple events hard to distinguish

Notes:

- For subgroup analysis (heterogeneity in effect size), see (Harper and Bruckner, 2017)
- To interpret with control variables, see (Bottomley et al., 2019)

Interrupted time series should be used when

1. longitudinal data (outcome over time - observations before and after the intervention)
2. full population was affected at one specific point in time (or can be stacked based on intervention)

In each ITS framework, there can be 4 possible scenarios of outcome after an intervention

- No effects
- Immediate effect

- Sustained (long-term) effect (smooth)
- Both immediate and sustained effect

$$Y = \beta_0 + \beta_1 T + \beta_2 D + \beta_3 P + \epsilon$$

where

- Y is the outcome variable
  - β_0 is the baseline level of the outcome
- T is the time variable (e.g., days, weeks, etc.) passed from the start of the observation period
  - β_1 is the slope of the line before the intervention
- D is the treatment variable where 1 is after the intervention and 0 is before the intervention.
  - β_2 is the **immediate effect** after the intervention
- P is the time variable indicating time passed since the intervention (before the intervention, the value is set to 0) (to examine the sustained effect).
  - β_3 is the **sustained effect** = difference between the slope of the line prior to the intervention and the slope of the line subsequent to the intervention

Example

Create a fictitious dataset where we know the true data generating process

$$Outcome = 10 \times time + 20 \times treatment + 25 \times timesincetreatment + noise$$

```
# number of days
n = 365


# intervention at day
interven = 200

# time index from 1 to 365
time = c(1:n)

# treatment variable: before internvation = day 1 to 200,
# after intervention = day 201 to 365
treatment = c(rep(0, interven), rep(1, n - interven))
```

```r
# time since treatment
timesincetreat = c(rep(0, interven), c(1:(n - interven)))

# outcome
outcome = 10 + 15 * time + 20 * treatment +
    25 * timesincetreat + rnorm(n, mean = 0, sd = 1)

df = data.frame(outcome, time, treatment, timesincetreat)

head(df, 10)
#>      outcome time treatment timesincetreat
#> 1   25.63968    1         0              0
#> 2   40.21753    2         0              0
#> 3   54.73614    3         0              0
#> 4   68.88212    4         0              0
#> 5   85.28696    5         0              0
#> 6   99.86008    6         0              0
#> 7  115.29658    7         0              0
#> 8  129.46741    8         0              0
#> 9  144.44288    9         0              0
#> 10 160.20887   10         0              0
```

Visualize

```r
plot(df$time, df$outcome)

# intervention date
abline(v = interven, col = "blue")

# regression line
ts <- lm(outcome ~ time + treatment + timesincetreat, data = df)
lines(df$time, ts$fitted.values, col = "red")
```

```
summary(ts)
#>
#> Call:
#> lm(formula = outcome ~ time + treatment + timesincetreat, data = df)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -3.7817 -0.6680  0.0494  0.6622  2.8242
#>
#> Coefficients:
#>                 Estimate Std. Error  t value Pr(>|t|)
#> (Intercept)     9.773455   0.149426    65.41   <2e-16 ***
#> time           15.002471   0.001289 11636.71   <2e-16 ***
#> treatment      19.749770   0.221593    89.13   <2e-16 ***
#> timesincetreat 24.996943   0.002150 11626.81   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.053 on 361 degrees of freedom
#> Multiple R-squared:      1,  Adjusted R-squared:      1
#> F-statistic: 8.604e+08 on 3 and 361 DF,  p-value: < 2.2e-16
```

Interpretation

- Time coefficient shows before-intervention outcome trend. Positive and significant, indicating a rising trend. Every day adds 15 points.
- The treatment coefficient shows the **immediate** increase in outcome. **Im-**

mediate effect is positive and significant, increasing outcome by 20 points.

- The time since treatment coefficient reflects a change in trend subsequent to the intervention. The **sustained effect** is positive and statistically significant, showing that the outcome increases by 25 points per day after the intervention.

See Lee Rodgers et al. (2014) for suggestions

Plot of counterfactual

```
# treatment prediction
pred <- predict(ts, df)

# counterfactual dataset
new_df <-
    as.data.frame(cbind(
        time = time,
        # treatment = 0 means counterfactual
        treatment = rep(0, n),
        # time since treatment = 0 means counterfactual
        timesincetreat = rep(0)
    ))

# counterfactual predictions
pred_cf <- predict(ts, new_df)

# plot
plot(
    outcome,
    col = gray(0.2, 0.2),
    pch = 19,
    xlim  = c(1,365),
    ylim = c(0, 10000),
    xlab = "xlab",
    ylab = "ylab"
)

# regression line before treatment
lines(rep(1:interven), pred[1:interven], col = "blue", lwd = 3)

# regression line after treatment
lines(rep((interven + 1):n), pred[(interven + 1):n],
      col = "blue", lwd = 3)

# regression line after treatment (counterfactual)
```

```
lines(
    rep(interven:n),
    pred_cf[(interven):n],
    col = "yellow",
    lwd = 3,
    lty = 5
)

abline(v = interven, col = "red", lty = 2)
```

Possible threats to the validity of interrupted time series analysis (Baicker and Svoronos, 2019)

- Delayed effects (Rodgers et al., 2005) (may have to make assess some time after the intervention - do not assess the immediate dates).
- Other confounding events (Linden and Yarnold, 2016, Linden (2017))
- Intervention is introduced but later withdrawn (Linden, 2015)
- Autocorrelation (for every time series data): might cause underestimation in the standard errors (i.e., overestimating the statistical significance of the treatment effect)
- Regression to the mean: after a the short-term shock to the outcome, individuals can revert back to their initial states.
- Selection bias: only certain individuals are affected by the treatment (could use a Multiple Groups).

35.1 Autocorrelation

Assess autocorrelation from residual

```
# simple regression on time
simple_ts <- lm(outcome ~ time, data = df)

plot(resid(simple_ts))
```

```
# alternatively
acf(resid(simple_ts))
```

This is not the best example since I created this dataset. But when residuals do have autocorrelation, you should not see any patterns (i.e., points should be randomly distributed on the plot)

To formally test for autocorrelation, we can use the Durbin-Watson test

```
lmtest::dwtest(df$outcome ~ df$time)
#>
#>  Durbin-Watson test
#>
#> data:  df$outcome ~ df$time
#> DW = 0.00037599, p-value < 2.2e-16
#> alternative hypothesis: true autocorrelation is greater than 0
```

From the p-value, we know that there is autocorrelation in the time series

A solution to this problem is to use more advanced time series analysis (e.g., ARIMA - coming up in the book) to adjust for seasonality and other dependency.

```
forecast::auto.arima(df$outcome, xreg = as.matrix(df[,-1]))
#> Series: df$outcome
#> Regression with ARIMA(0,0,0) errors
#>
#> Coefficients:
#>       intercept     time  treatment  timesincetreat
#>          9.7735  15.0025    19.7498         24.9969
#> s.e.     0.1486   0.0013     0.2204          0.0021
#>
#> sigma^2 = 1.108:  log likelihood = -534.63
```

```
#> AIC=1079.26   AICc=1079.42   BIC=1098.76
```

35.2 Multiple Groups

When you suspect that you might have confounding events or selection bias, you can add a control group that did not experience the treatment (very much similar to Difference-in-differences)

The model then becomes

$$\begin{aligned} Y = \beta_0 &+ \beta_1 time + \beta_2 treatment + \beta_3 \times timesincetreat \\ &+ \beta_4 group + \beta_5 group \times time + \beta_6 group \times treatment \\ &+ \beta_7 group \times timesincetreat \end{aligned}$$

where

- Group = 1 when the observation is under treatment and 0 under control
- β_4 = baseline difference between the treatment and control group
- β_5 = slope difference between the treatment and control group before treatment
- β_6 = baseline difference between the treatment and control group associated with the treatment.
- β_7 = difference between the sustained effect of the treatment and control group after the treatment.

C. OTHER CONCERNS

Chapter 36

Endogeneity

Refresher

A general model framework

$$\mathbf{Y} = \mathbf{X} \ +$$

where

- $\mathbf{Y} = n \times 1$
- $\mathbf{X} = n \times k$
- $\beta = k \times 1$
- $\epsilon = n \times 1$

Then, OLS estimates of coefficients are

$$\begin{aligned}
\hat{\beta}_{OLS} &= (\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'\mathbf{Y}) \\
&= (\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'(\mathbf{X} \ + \ )) \\
&= (\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'\mathbf{X})\beta + (\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}' \ ) \\
\hat{\beta}_{OLS} &\to \beta + (\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}' \ )
\end{aligned}$$

To have unbiased estimates, we have to get rid of the second part $(\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}' \ )$

There are 2 conditions to achieve unbiased estimates:

1. $E(\epsilon|X) = 0$ (This is easy, putting an intercept can solve this issue)
2. $Cov(\mathbf{X}, \epsilon) = 0$ (This is the hard part)

We only care about omitted variable

Usually, the problem will stem Omitted Variables Bias, but we only care about omitted variable bias when

1. Omitted variables correlate with the variables we care about (X). If OMV does not correlate with X, we don't care, and random assignment makes this correlation goes to 0)
2. Omitted variables correlates with outcome/ dependent variable

There are more types of endogeneity listed below.

Types of endogeneity (See Hill et al. (2021) for a review in management):

1. Endogenous Treatment

- Omitted Variables Bias
  - Motivation
  - Ability/talent
  - Self-selection
- Feedback Effect (Simultaneity): also known as bidirectionality
- Reverse Causality: Subtle difference from Simultaneity: Technically, two variables affect each other sequentially, but in a big enough time frame, (e.g., monthly, or yearly), our coefficient will be biased just like simultaneity.
- Measurement Error

2. Endogenous Sample Selection

To deal with this problem, we have a toolbox (that has been mentioned in previous chapter **??**)

Using control variables in regression is a "selection on observables" identification strategy.

In other words, if you believe you have an omitted variable, and you can measure it, including it in the regression model solves your problem. These uninterested variables are called control variables in your model.

However, this is rarely the case (because the problem is we don't have their measurements). Hence, we need more elaborate methods:

- Endogenous Treatment
- Endogenous Sample Selection

Before we get to methods that deal with bias arises from omitted variables, we consider cases where we do have measurements of a variable, but there is measurement error (bias).

36.1 Endogenous Treatment

36.1.1 Measurement Error

- Data error can stem from
  - Coding errors
  - Reporting errors

Two forms of measurement error:

1. Random (stochastic) (indeterminate error) (Classical Measurement Errors): noise or measurement errors do not show up in a consistent or predictable way.
2. Systematic (determinate error) (Non-classical Measurement Errors): When measurement error is consistent and predictable across observations.
   1. Instrument errors (e.g., faulty scale) -> calibration or adjustment
   2. Method errors (e.g., sampling errors) -> better method development + study design
   3. Human errors (e.g., judgement)

Usually the systematic measurement error is a bigger issue because it introduces "bias" into our estimates, while random error introduces noise into our estimates

- Noise -> regression estimate to 0
- Bias -> can pull estimate to upward or downward.

36.1.1.1 Classical Measurement Errors

36.1.1.1.1 Right-hand side

- Right-hand side measurement error: When the measurement is in the covariates, then we have the endogeneity problem.

Say you know the true model is

$$Y_i = \beta_0 + \beta_1 X_i + u_i$$

But you don't observe X_i, but you observe

$$\tilde{X}_i = X_i + e_i$$

which is known as classical measurement errors where we **assume** e_i is uncorrelated with X_i (i.e., $E(X_i e_i) = 0$)

Then, when you estimate your observed variables, you have (substitute X_i with $\tilde{X}_i - e_i$):

$$\begin{aligned} Y_i &= \beta_0 + \beta_1(\tilde{X}_i - e_i) + u_i \\ &= \beta_0 + \beta_1\tilde{X}_i + u_i - \beta_1 e_i \\ &= \beta_0 + \beta_1\tilde{X}_i + v_i \end{aligned}$$

In words, the measurement error in X_i is now a part of the error term in the regression equation v_i. Hence, we have an endogeneity bias.

Endogeneity arises when

$$\begin{aligned} E(\tilde{X}_i v_i) &= E((X_i + e_i)(u_i - \beta_1 e_i)) \\ &= -\beta_1 Var(e_i) \neq 0 \end{aligned}$$

Since $\tilde{X}_i$ and e_i are positively correlated, then it leads to

- a negative bias in $\hat{\beta}_1$ if the true β_1 is positive
- a positive bias if β_1 is negative

In other words, measurement errors cause **attenuation bias**, which inter turn pushes the coefficient towards 0

As $Var(e_i)$ increases or $\frac{Var(e_i)}{Var(\tilde{X})} \to 1$ then e_i is a random (noise) and $\beta_1 \to 0$ (random variable $\tilde{X}$ should not have any relation to Y_i)

Technical note:

The size of the bias in the OLS-estimator is

$$\hat{\beta}_{OLS} = \frac{cov(\tilde{X}, Y)}{var(\tilde{X})} = \frac{cov(X + e, \beta X + u)}{var(X + e)}$$

then

$$plim\hat{\beta}_{OLS} = \beta\frac{\sigma_X^2}{\sigma_X^2 + \sigma_e^2} = \beta\lambda$$

where λ is **reliability** or signal-to-total variance ratio or attenuation factor

Reliability affect the extent to which measurement error attenuates $\hat{\beta}$. The attenuation bias is

$$\hat{\beta}_{OLS} - \beta = -(1 - \lambda)\beta$$

Thus, $\hat{\beta}_{OLS} < \beta$ (unless $\lambda = 1$, in which case we don't even have measurement error).

Note:

Data transformation worsen (magnify) the measurement error

$$y = \beta x + \gamma x^2 + \epsilon$$

then, the attenuation factor for $\hat{\gamma}$ is the square of the attenuation factor for $\hat{\beta}$ (i.e., $\lambda_{\hat{\gamma}} = \lambda^2_{\hat{\beta}}$)

Adding covariates increases attenuation bias

To fix classical measurement error problem, we can

1. Find estimates of either $\sigma^2_X, \sigma^2_\epsilon$ or λ from validation studies, or survey data.
2. Endogenous Treatment Use instrument Z correlated with X but uncorrelated with ϵ
3. Abandon your project

36.1.1.1.2 Left-hand side When the measurement is in the outcome variable, econometricians or causal scientists do not care because they still have an unbiased estimate of the coefficients (the zero conditional mean assumption is not violated, hence we don't have endogeneity). However, statisticians might care because it might inflate our uncertainty in the coefficient estimates (i.e., higher standard errors).

$$\tilde{Y} = Y + v$$

then the model you estimate is

$$\tilde{Y} = \beta X + u + v$$

Since v is uncorrelated with X, then $\hat{\beta}$ is consistently estimated by OLS

If we have measurement error in Y_i, it will pass through β_1 and go to u_i

36.1.1.2 Non-classical Measurement Errors

Relaxing the assumption that X and ϵ are uncorrelated

Recall the true model we have true estimate is

$$\hat{\beta} = \frac{cov(X + \epsilon, \beta X + u)}{var(X + \epsilon)}$$

then without the above assumption, we have

$$
\begin{aligned}
plim\hat{\beta} &= \frac{\beta(\sigma_X^2 + \sigma_{X\epsilon})}{\sigma_X^2 + \sigma_\epsilon^2 + 2\sigma_{X\epsilon}} \\
&= (1 - \frac{\sigma_\epsilon^2 + \sigma_{X\epsilon}}{\sigma_X^2 + \sigma_\epsilon^2 + 2\sigma_{X\epsilon}})\beta \\
&= (1 - b_{\epsilon\tilde{X}})\beta
\end{aligned}
$$

where $b_{\epsilon\tilde{X}}$ is the covariance between $\tilde{X}$ and ϵ (also the regression coefficient of a regression of ϵ on $\tilde{X}$)

Hence, the Classical Measurement Errors is just a special case of Non-classical Measurement Errors where $b_{\epsilon\tilde{X}} = 1 - \lambda$

So when $\sigma_{X\epsilon} = 0$ (Classical Measurement Errors), increasing this covariance $b_{\epsilon\tilde{X}}$ increases the covariance increases the attenuation factor if more than half of the variance in $\tilde{X}$ is measurement error, and decreases the attenuation factor otherwise. This is also known as **mean reverting measurement error** (Bound et al., 1989, Bound et al. (2001))

A general framework for both right-hand side and left-hand side measurement error is (Bound et al., 2001):

consider the true model

$$
\mathbf{Y} = \mathbf{X} \ +
$$

then

$$
\begin{aligned}
\hat{\beta} &= (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-\mathbf{1}}\tilde{\mathbf{X}}\tilde{\mathbf{Y}} \\
&= (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-\mathbf{1}}\tilde{\mathbf{X}}'(\tilde{\mathbf{X}} \ - \mathbf{U} \ + \mathbf{v} + \ ) \\
&= \ + (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-\mathbf{1}}\tilde{\mathbf{X}}'(-\mathbf{U} \ + \mathbf{v} + \ ) \\
plim\hat{\beta} &= \beta + plim(\tilde{X}'\tilde{X})^{-1}\tilde{X}'(-U\beta + v) \\
&= \beta + plim(\tilde{X}'\tilde{X})^{-1}\tilde{X}'W \begin{bmatrix} -\beta \\ 1 \end{bmatrix}
\end{aligned}
$$

Since we collect the measurement errors in a matrix $W = [U|v]$, then

$$
(-U\beta + v) = W \begin{bmatrix} -\beta \\ 1 \end{bmatrix}
$$

Hence, in general, biases in the coefficients β are regression coefficients from regressing the measurement errors on the mis-measured $\tilde{X}$

Notes:

- Instrumental Variable can help fix this problem
- There can also be measurement error in dummy variables and you can still use Instrumental Variable to fix it.

36.1.1.3 Solution to Measurement Errors

36.1.1.3.1 Correlation

$$P(\rho|data) = \frac{P(data|\rho)P(\rho)}{P(data)}$$

$$\text{Posterior Probability} \propto \text{Likelihood} \times \text{Prior Probability}$$

where

- ρ is a correlation coefficient
- $P(data|\rho)$ is the likelihood function evaluated at ρ
- $P(\rho)$ prior probability
- $P(data)$ is the normalizing constant

With sample correlation coefficient r:

$$r = \frac{S_{xy}}{\sqrt{S_{xx}S_{yy}}}$$

Then the posterior density approximation of ρ is (Schisterman et al., 2003, pp.3)

$$P(\rho|x,y) \propto P(\rho)\frac{(1-\rho^2)^{(n-1)/2}}{(1-\rho \times r)^{n-(3/2)}}$$

where

- $\rho = \tanh \xi$ where $\xi \sim N(z, 1/n)$
- $r = \tanh z$

Then the posterior density follow a normal distribution where

Mean

$$\mu_{posterior} = \sigma^2_{posterior} \times (n_{prior} \times \tanh^{-1} r_{prior} + n_{likelihood} \times \tanh^{-1} r_{likelihood})$$

variance

$$\sigma^2_{posterior} = \frac{1}{n_{prior} + n_{Likelihood}}$$

To simplify the integration process, we choose prior that is

$$P(\rho) \propto (1-\rho^2)^c$$

where

- c is the weight the prior will have in estimation (i.e., $c = 0$ if no prior info, hence $P(\rho) \propto 1$)

Example:

Current study: $r_{xy} = 0.5, n = 200$

Previous study: $r_{xy} = 0.2765, (n = 50205)$

Combining two, we have the posterior following a normal distribution with the **variance** of

$$\sigma^2_{posterior} = \frac{1}{n_{prior} + n_{Likelihood}} = \frac{1}{200 + 50205} = 0.0000198393$$

Mean

$$\begin{aligned}\mu_{Posterior} &= \sigma^2_{Posterior} \times (n_{prior} \times \tanh^{-1} r_{prior} + n_{likelihood} \times \tanh^{-1} r_{likelihood}) \\ &= 0.0000198393 \times (50205 \times \tanh^{-1} 0.2765 + 200 \times \tanh^{-1} 0.5) \\ &= 0.2849415\end{aligned}$$

Hence, $Posterior \sim N(0.691, 0.0009)$, which means the correlation coefficient is $\tanh(0.691) = 0.598$ and 95% CI is

$$\mu_{posterior} \pm 1.96 \times \sqrt{\sigma^2_{Posterior}} = 0.2849415 \pm 1.96 \times (0.0000198393)^{1/2} = (0.2762115, 0.2936714)$$

Hence, the interval for posterior ρ is $(0.2693952, 0.2855105)$

If future authors suspect that they have

1. Large sampling variation
2. Measurement error in either measures in the correlation, which attenuates the relationship between the two variables

Applying this Bayesian correction can give them a better estimate of the correlation between the two.

To implement this calculation in R, see below

```r
n_new              <- 200
r_new              <- 0.5
alpha              <- 0.05

update_correlation <- function(n_new, r_new, alpha) {
    n_meta             <- 50205
    r_meta             <- 0.2765

    # Variance
    var_xi         <- 1 / (n_new + n_meta)
    format(var_xi, scientific = FALSE)

    # mean
    mu_xi          <- var_xi * (n_meta * atanh(r_meta) + n_new * (atanh(r_new)))
    format(mu_xi, scientific  = FALSE)

    # confidence interval
    upper_xi       <- mu_xi + qnorm(1 - alpha / 2) * sqrt(var_xi)
    lower_xi       <- mu_xi - qnorm(1 - alpha / 2) * sqrt(var_xi)

    # rho
    mean_rho       <- tanh(mu_xi)
    upper_rho      <- tanh(upper_xi)
    lower_rho      <- tanh(lower_xi)

    # return a list
    return(
        list(
            "mu_xi" = mu_xi,
            "var_xi" = var_xi,
            "upper_xi" = upper_xi,
            "lower_xi" = lower_xi,
            "mean_rho" = mean_rho,
            "upper_rho" = upper_rho,
            "lower_rho" = lower_rho
        )
    )
}




# Old confidence interval
r_new + qnorm(1 - alpha / 2) * sqrt(1/n_new)
#> [1] 0.6385904
```

```
r_new - qnorm(1 - alpha / 2) * sqrt(1/n_new)
#> [1] 0.3614096

testing = update_correlation(n_new = n_new, r_new = r_new, alpha = alpha)

# Updated rho
testing$mean_rho
#> [1] 0.2774723

# Updated confidence interval
testing$upper_rho
#> [1] 0.2855105
testing$lower_rho
#> [1] 0.2693952
```

36.1.2 Simultaneity

- When independent variables (X's) are jointly determined with the dependent variable Y, typically through an equilibrium mechanism, violates the second condition for causality (i.e., temporal order).
- Examples: quantity and price by demand and supply, investment and productivity, sales and advertisement

General Simultaneous (Structural) Equations

$$
\begin{aligned}
Y_i &= \beta_0 + \beta_1 X_i + u_i \\
X_i &= \alpha_0 + \alpha_1 Y_i + v_i
\end{aligned}
$$

Hence, the solutions are

$$
\begin{aligned}
Y_i &= \frac{\beta_0 + \beta_1 \alpha_0}{1 - \alpha_1 \beta_1} + \frac{\beta_1 v_i + u_i}{1 - \alpha_1 \beta_1} \\
X_i &= \frac{\alpha_0 + \alpha_1 \beta_0}{1 - \alpha_1 \beta_1} + \frac{v_i + \alpha_1 u_i}{1 - \alpha_1 \beta_1}
\end{aligned}
$$

If we run only one regression, we will have biased estimators (because of **simultaneity bias**):

$$
\begin{aligned}
Cov(X_i, u_i) &= Cov(\frac{v_i + \alpha_1 u_i}{1 - \alpha_1 \beta_1}, u_i) \\
&= \frac{\alpha_1}{1 - \alpha_1 \beta_1} Var(u_i)
\end{aligned}
$$

In an even more general model

$$\begin{cases} Y_i = \beta_0 + \beta_1 X_i + \beta_2 T_i + u_i \\ X_i = \alpha_0 + \alpha_1 Y_i + \alpha_2 Z_i + v_i \end{cases}$$

where

- X_i, Y_i are **endogenous** variables determined within the system
- T_i, Z_i are **exogenous** variables

Then, the reduced form of the model is

$$\begin{cases} Y_i = \dfrac{\beta_0 + \beta_1\alpha_0}{1-\alpha_1\beta_1} + \dfrac{\beta_1\alpha_2}{1-\alpha_1\beta_1} Z_i + \dfrac{\beta_2}{1-\alpha_1\beta_1} T_i + \tilde{u}_i \\ \quad = B_0 + B_1 Z_i + B_2 T_i + \tilde{u}_i \\ X_i = \dfrac{\alpha_0 + \alpha_1\beta_0}{1-\alpha_1\beta_1} + \dfrac{\alpha_2}{1-\alpha_1\beta_1} Z_i + \dfrac{\alpha_1\beta_2}{1-\alpha_1\beta_1} T_i + \tilde{v}_i \\ \quad = A_0 + A_1 Z_i + A_2 T_i + \tilde{v}_i \end{cases}$$

Then, now we can get consistent estimates of the reduced form parameters

And to get the original parameter estimates

$$\begin{aligned} \frac{B_1}{A_1} &= \beta_1 \\ B_2(1 - \frac{B_1 A_2}{A_1 B_2}) &= \beta_2 \\ \frac{A_2}{B_2} &= \alpha_1 \\ A_1(1 - \frac{B_1 A_2}{A_1 B_2}) &= \alpha_2 \end{aligned}$$

Rules for Identification

Order Condition (necessary but not sufficient)

$$K - k \geq m - 1$$

where

- M = number of endogenous variables in the model
- K = number of exogenous variables int he model
- m = number of endogenous variables in a given
- k = is the number of exogenous variables in a given equation

This is actually the general framework for instrumental variables

36.1.3 Endogenous Treatment Solutions

Using the OLS estimates as a reference point

```
library(AER)
library(REndo)
set.seed(421)
data("CASchools")
school <- CASchools
school$stratio <- with(CASchools, students / teachers)
m1.ols <-
    lm(read ~ stratio + english + lunch
       + grades + income + calworks + county,
       data = school)
summary(m1.ols)$coefficients[1:7,]
#>                 Estimate Std. Error     t value      Pr(>|t|)
#> (Intercept) 683.45305948 9.56214469  71.4748711 3.011667e-218
#> stratio      -0.30035544 0.25797023  -1.1643027  2.450536e-01
#> english      -0.20550107 0.03765408  -5.4576041  8.871666e-08
#> lunch        -0.38684059 0.03700982 -10.4523759  1.427370e-22
#> gradesKK-08  -1.91291321 1.35865394  -1.4079474  1.599886e-01
#> income        0.71615378 0.09832843   7.2832829  1.986712e-12
#> calworks     -0.05273312 0.06154758  -0.8567863  3.921191e-01
```

36.1.3.1 Instrumental Variable

[A3a] requires ϵ_i to be uncorrelated with $\mathbf{x}_i$

Assume A1 , A2, A5

$$plim(\hat{\beta}_{OLS}) = \beta + [E(\mathbf{x_i'x_i})]^{-1}E(\mathbf{x_i'}\epsilon_i)$$

[A3a] is the weakest assumption needed for OLS to be **consistent**

A3 fails when x_{ik} is correlated with ϵ_i

- Omitted Variables Bias: ϵ_i includes any other factors that may influence the dependent variable (linearly)
- Simultaneity Demand and prices are simultaneously determined.
- Endogenous Sample Selection we did not have iid sample
- Measurement Error

Note

- Omitted Variable: an omitted variable is a variable, omitted from the model (but is in the ϵ_i) and unobserved has predictive power towards the outcome.
- Omitted Variable Bias: is the bias (and inconsistency when looking at large sample properties) of the OLS estimator when the omitted variable.

- We cam have both positive and negative selection bias (it depends on what our story is)

The **structural equation** is used to emphasize that we are interested understanding a **causal relationship**

$$y_{i1} = \beta_0 + \mathbf{z}_i 1\beta_1 + y_{i2}\beta_2 + \epsilon_i$$

where

- y_{it} is the outcome variable (inherently correlated with ϵ_i)
- y_{i2} is the endogenous covariate (presumed to be correlated with ϵ_i)
- β_1 represents the causal effect of y_{i2} on y_{i1}
- $\mathbf{z}_{i1}$ is exogenous controls (uncorrelated with ϵ_i) ($E(z'_{1i}\epsilon_i) = 0$)

OLS is an inconsistent estimator of the causal effect β_2

If there was no endogeneity

- $E(y'_{i2}\epsilon_i) = 0$
- the exogenous variation in y_{i2} is what identifies the causal effect

If there is endogeneity

- Any wiggle in y_{i2} will shift simultaneously with ϵ_i

$$plim(\hat{\beta}_{OLS}) = \beta + [E(\mathbf{x'_i x_i})]^{-1} E(\mathbf{x'_i}\epsilon_i)$$

where

- β is the causal effect
- $[E(\mathbf{x'_i x_i})]^{-1} E(\mathbf{x'_i}\epsilon_i)$ is the endogenous effect

Hence $\hat{\beta}_{OLS}$ can be either more positive and negative than the true causal effect.

Motivation for **Two Stage Least Squares (2SLS)**

$$y_{i1} = \beta_0 + \mathbf{z}_{i1}\beta_1 + y_{i2}\beta_2 + \epsilon_i$$

We want to understand how movement in y_{i2} effects movement in y_{i1}, but whenever we move y_{i2}, ϵ_i also moves.

Solution
We need a way to move y_{i2} independently of ϵ_i, then we can analyze the response in y_{i1} as a causal effect

- Find an **instrumental variable(s)** z_{i2}
  - Instrument **Relevance**: when** z_{i2} moves then y_{i2} also moves
  - Instrument **Exogeneity**: when z_{i2} moves then ϵ_i does not move.

- z_{i2} is the **exogenous variation that identifies** the causal effect β_2

Finding an Instrumental variable:

- Random Assignment: + Effect of class size on educational outcomes: instrument is initial random
- Relation's Choice + Effect of Education on Fertility: instrument is parent's educational level
- Eligibility + Trade-off between IRA and 401K retirement savings: instrument is 401k eligibility

Example

Return to College

- education is correlated with ability - endogenous
- **Near 4year** as an instrument
  - Instrument Relevance: when **near** moves then education also moves
  - Instrument Exogeneity: when **near** moves then ϵ_i does not move.
- Other potential instruments; near a 2-year college. Parent's Education. Owning Library Card

$$y_{i1} = \beta_0 + \mathbf{z}_{i1}\beta_1 + y_{i2}\beta_2 + \epsilon_i$$

First Stage (Reduced Form) Equation:

$$y_{i2} = \pi_0 + \mathbf{z_{i1\ 1}} + \mathbf{z_{i2\ 2}} + v_i$$

where

- $\pi_0 + \mathbf{z_{i1\ 1}} + \mathbf{z_{i2\ 2}}$ is exogenous variation v_i is endogenous variation

This is called a **reduced form equation**

- Not interested in the causal interpretation of π_1 or π_2
- A linear projection of z_{i1} and z_{i2} on y_{i2} (simple correlations)
- The projections π_1 and π_2 guarantee that $E(z'_{i1}v_i) = 0$ and $E(z'_{i2}v_i) = 0$

Instrumental variable z_{i2}

- **Instrument Relevance**: $\pi_2 \neq 0$
- **Instrument Exogeneity**: $E(\mathbf{z_{i2\ i}}) = 0$

Moving only the exogenous part of y_i2 is moving

$$\tilde{y}_{i2} = \pi_0 + \mathbf{z_{i1\ 1}} + \mathbf{z_{i2\ 2}}$$

two Stage Least Squares (2SLS)

$$y_{i1} = \beta_0 + \mathbf{z_{i1}}_{\ \mathbf{1}} + y_{i2}\beta_2 + \epsilon_i$$

$$y_{i2} = \pi_0 + \mathbf{z_{i2}}_{\ \mathbf{2}} + \mathbf{v_i}$$

Equivalently,

$$y_{i1} = \beta_0 + \mathbf{z_{i1}}\beta_1 + \tilde{y}_{i2}\beta_2 + u_i \tag{36.1}$$

where

- $\tilde{y}_{i2} = \pi_0 + \mathbf{z_{i2}}_{\ \mathbf{2}}$
- $u_i = v_i\beta_2 + \epsilon_i$

The (36.1) holds for A1, A5

- A2 holds if the instrument is relevant $\pi_2 \neq 0$ + $y_{i1} = \beta_0 + \mathbf{z_{i1}}_{\ \mathbf{1}} + (\ _{\mathbf{0}} + \mathbf{z_{i1}}_{\ \mathbf{1}} + \mathbf{z_{i2}}_{\ \mathbf{2}})\beta_2 + u_i$
- [A3a] holds if the instrument is exogenous $E(\mathbf{z}_{i2}\epsilon_i) = 0$

$$\begin{aligned} E(\tilde{y}'_{i2}u_i) &= E((\pi_0 + \mathbf{z_{i1}}_{\ \mathbf{1}} + \mathbf{z_{i2}})(v_i\beta_2 + \epsilon_i)) \\ &= E((\pi_0 + \mathbf{z_{i1}}_{\ \mathbf{1}} + \mathbf{z_{i2}})(\epsilon_i)) \\ &= E(\epsilon_i)\pi_0 + E(\epsilon_i z_{i1})\pi_1 + E(\epsilon_i z_{i2}) \\ &= 0 \end{aligned}$$

Hence, (36.1) is consistent

The 2SLS Estimator
1. Estimate the first stage using OLS

$$y_{i2} = \pi_0 + \mathbf{z_{i2}}_{\ \mathbf{2}} + \mathbf{v_i}$$

and obtained estimated value $\hat{y}_{i2}$

2. Estimate the altered equation using OLS

$$y_{i1} = \beta_0 + \mathbf{z_{i1}}_{\ \mathbf{1}} + \hat{y}_{i2}\beta_2 + \epsilon_i$$

Properties of the 2SLS Estimator

- Under A1, A2, [A3a] (for z_{i1}), A5 and if the instrument satisfies the following two conditions, + **Instrument Relevance**: $\pi_2 \neq 0$ + **Instrument Exogeneity**: $E(\mathbf{z}'_{i2}\epsilon_i) = 0$ then the 2SLS estimator is consistent

- Can handle more than one endogenous variable and more than one instrumental variable

$$
\begin{aligned}
y_{i1} &= \beta_0 + z_{i1}\beta_1 + y_{i2}\beta_2 + y_{i3}\beta_3 + \epsilon_i \\
y_{i2} &= \pi_0 + z_{i1}\pi_1 + z_{i2}\pi_2 + z_{i3}\pi_3 + z_{i4}\pi_4 + v_{i2} \\
y_{i3} &= \gamma_0 + z_{i1}\gamma_1 + z_{i2}\gamma_2 + z_{i3}\gamma_3 + z_{i4}\gamma_4 + v_{i3}
\end{aligned}
$$

```
+ **IV estimator**: one endogenous variable with a single instrument
+ **2SLS estimator**: one endogenous variable with multiple instruments
+ **GMM estimator**: multiple endogenous variables with multiple instruments
```

- Standard errors produced in the second step are not correct
  - Because we do not know $\tilde{y}$ perfectly and need to estimate it in the firs step, we are introducing additional variation
  - We did not have this problem with FGLS because "the first stage was orthogonal to the second stage." This is generally not true for most multi-step procedure.
  - If A4 does not hold, need to report robust standard errors.
- 2SLS is less efficient than OLS and will always have larger standard errors.
  - First, $Var(u_i) = Var(v_i\beta_2 + \epsilon_i) > Var(\epsilon_i)$
  - Second, $\hat{y}_{i2}$ is generally highly collinear with $\mathbf{z}_{i1}$
- The number of instruments need to be at least as many or more the number of endogenous variables.

Note

- 2SLS can be combined with FGLS to make the estimator more efficient: You have the same first-stage, and in the second-stage, instead of using OLS, you can use FLGS with the weight matrix $\hat{w}$
- Generalized Method of Moments can be more efficient than 2SLS.
- In the second-stage of 2SLS, you can also use MLE, but then you are making assumption on the distribution of the outcome variable, the endogenous variable, and their relationship (joint distribution).

36.1.3.1.1 Testing Assumptions

1. Endogeneity Test: Is y_{i2} truly endogenous (i.e., can we just use OLS instead of 2SLS)?
2. Exogeneity (Cannot always test and when you can it might not be informative)

3. Relevancy (need to avoid "weak instruments")

36.1.3.1.1.1 Endogeneity Test

- 2SLS is generally so inefficient that we may prefer OLS if there is not much endogeneity
- Biased but inefficient vs efficient but biased
- Want a sense of "how endogenous" y_{i2} is
  - if "very" endogenous - should use 2SLS
  - if not "very" endogenous - perhaps prefer OLS

Invalid Test of Endogeneity: y_{i2} is endogenous if it is correlated with ϵ_i,

$$\epsilon_i = \gamma_0 + y_{i2}\gamma_1 + error_i$$

where $\gamma_1 \neq 0$ implies that there is endogeneity

- ϵ_i is not observed, but using the residuals

$$e_i = \gamma_0 + y_{i2}\gamma_1 + error_i$$

is **NOT** a valid test of endogeneity + The OLS residual, e is mechanically uncorrelated with y_{i2} (by FOC for OLS) + In every situation, γ_1 will be essentially 0 and you will never be able to reject the null of no endogeneity

Valid test of endogeneity

- If y_{i2} is not endogenous then ϵ_i and v are uncorrelated

$$\begin{aligned} y_{i1} &= \beta_0 + \mathbf{z}_{i1}\beta_1 + y_{i2}\beta_2 + \epsilon_i \\ y_{i2} &= \pi_0 + \mathbf{z}_{i1}\pi_1 + z_{i2}\pi_2 + v_i \end{aligned}$$

Variable Addition test: include the first stage residuals as an additional variable,

$$y_{i1} = \beta_0 + \mathbf{z}_{i1}\beta_1 + y_{i2}\beta_2 + \hat{v}_i\theta + error_i$$

Then the usual t-test of significance is a valid test to evaluate the following hypothesis. **note** this test requires your instrument to be valid instrument.

$$\begin{aligned} H_0 &: \theta = 0 \quad \text{(not endogenous)} \\ H_1 &: \theta \neq 0 \quad \text{(endogenous)} \end{aligned}$$

36.1.3.1.1.2 Exogeneity Why exogeneity matter?

$$E(\mathbf{z}'_{i2}\epsilon_i) = 0$$

- If [A3a] fails - 2SLS is also inconsistent
- If instrument is not exogenous, then we need to find a new one.
- Similar to Endogeneity Test, when there is a single instrument

$$e_i = \gamma_0 + \mathbf{z}_{i2}\gamma_1 + error_i$$
$$H_0 : \gamma_1 = 0$$

is **NOT** a valid test of endogeneity

- the OLS residual, e is mechanically uncorrelated with z_{i2}: $\hat{\gamma}_1$ will be essentially 0 and you will never be able to determine if the instrument is endogenous.

Solution

Testing Instrumental Exogeneity in an Over-identified Model

- When there is more than one exogenous instrument (per endogenous variable), we can test for instrument exogeneity.
  - When we have multiple instruments, the model is said to be over-identified.
  - Could estimate the same model several ways (i.e., can identify/ estimate β_1 more than one way)
- Idea behind the test: if the controls and instruments are truly exogenous then OLS estimation of the following regression,

$$\epsilon_i = \gamma_0 + \mathbf{z}_{i1}\gamma_1 + \mathbf{z}_{i2}\gamma_2 + error_i$$

should have a very low R^2

- if the model is **just identified** (one instrument per endogenous variable) then the $R^2 = 0$

Steps:

(1) Estimate the structural equation by 2SLS (using all available instruments) and obtain the residuals e

(2) Regress e on all controls and instruments and obtain the R^2

(3) Under the null hypothesis (all IV's are uncorrelated), $nR^2 \sim \chi^2(q)$, where q is the number of instrumental variables minus the number of endogenous variables

- if the model is just identified (one instrument per endogenous variable) then q = 0, and the distribution under the null collapses.

low p-value means you reject the null of exogenous instruments. Hence you would like to have high p-value in this test.

Pitfalls for the Overid test

- the overid test is essentially compiling the following information.
  - Conditional on first instrument being exogenous is the other instrument exogenous?
  - Conditional on the other instrument being exogenous, is the first instrument exogenous?
- If all instruments are endogenous than neither test will be valid
- really only useful if one instrument is thought to be truly exogenous (randomly assigned). even f you do reject the null, the test does not tell you which instrument is exogenous and which is endogenous.

Result	Implication
reject the null	you can be pretty sure there is an endogenous instrument, but don't know which one.
fail to reject	could be either (1) they are both exogenous, (2) they are both endogenous.

36.1.3.1.1.3 Relevancy Why Relevance matter?

$$\pi_2 \neq 0$$

- used to show A2 holds
  - If $\pi_2 = 0$ (instrument is not relevant) then A2 fails - perfect multicollinearity
  - If π_2 is close to 0 (**weak instrument**) then there is near perfect multicollinearity - 2SLS is highly inefficient (Large standard errors).
- A weak instrument will exacerbate any inconsistency due to an instrument being (even slightly) endogenous.
  - In the simple case with no controls and a single endogenous variable and single instrumental variable,

$$plim(\hat{\beta}_{2_{2SLS}}) = \beta_2 + \frac{E(z_{i2}\epsilon_i)}{E(z_{i2}y_{i2})}$$

Testing Weak Instruments

- can use t-test (or F-test for over-identified models) in the first stage to determine if there is a weak instrument problem.
- (Stock and Yogo, 2002, Stock and Yogo (2005a)): a statistical rejection of the null hypothesis in the first stage at the 5% (or even 1%) level is not enough to insure the instrument is not weak
  - Rule of Thumb: need a F-stat of at least 10 (or a t-stat of at least 3.2) to reject the null hypothesis that the instrument is weak.

Summary of the 2SLS Estimator

$$y_{i1} = \beta_0 + \mathbf{z}_{i1}\beta_1 + y_{i2}\beta_2 + \epsilon_i$$
$$y_{i2} = \pi_0 + \mathbf{z_{i1}}_{\ \mathbf{1}} + \mathbf{z_{i2}}_{\ \mathbf{2}} + v_i$$

- when [A3a] does not hold

$$E(y_{i2}'\epsilon_i) \neq 0$$

- Then the OLS estimator is no longer unbiased or consistent.
- If we have valid instruments $\mathbf{z}_{i2}$
- Relevancy (need to avoid "weak instruments"): $\pi_2 \neq 0$ Then the 2SLS estimator is consistent under A1, A2, [A5a], and the above two conditions.
  - If A4 also holds, then the usual standard errors are valid.
  - If A4 does not hold then use the robust standard errors.

$$y_{i1} = \beta_0 + \mathbf{z}_{i1}\beta_1 + y_{i2}\beta_2 + \epsilon_i$$
$$y_{i2} = \pi_0 + \mathbf{z_{i1}}_{\ \mathbf{1}} + \mathbf{z_{i2}}_{\ \mathbf{2}} + v_i$$

- When [A3a] does hold

$$E(y_{i2}'\epsilon_i) = 0$$

and we have valid instruments, then both the OLS and 2SLS estimators are consistent.

- The OLS estimator is always more efficient
- can use the variable addition test to determine if 2SLS is need (A3a does hold) or if OLS is valid (A3a does not hold)

Sometimes we can test the assumption for instrument to be valid:

- Exogeneity : Only table when there are more instruments than endogenous variables.
- Relevancy (need to avoid "weak instruments"): Always testable, need the F-stat to be greater than 10 to rule out a weak instrument

Application

Expenditure as observed instrument

```
m2.2sls <-
    ivreg(
        read ~ stratio + english + lunch
        + grades + income + calworks + county |

            expenditure + english + lunch
        + grades + income + calworks + county ,
        data = school
    )
summary(m2.2sls)$coefficients[1:7,]
#>                 Estimate  Std. Error     t value      Pr(>|t|)
#> (Intercept) 700.47891593 13.58064436  51.5792106 8.950497e-171
#> stratio      -1.13674002  0.53533638  -2.1234126  3.438427e-02
#> english      -0.21396934  0.03847833  -5.5607753  5.162571e-08
#> lunch        -0.39384225  0.03773637 -10.4366757  1.621794e-22
#> gradesKK-08  -1.89227865  1.37791820  -1.3732881  1.704966e-01
#> income        0.62487986  0.11199008   5.5797785  4.668490e-08
#> calworks     -0.04950501  0.06244410  -0.7927892  4.284101e-01
```

36.1.3.1.2 Checklist

1. Regress the dependent variable on the instrument (reduced form). Since under OLS, we have unbiased estimate, the coefficient estimate should be significant (make sure the sign makes sense)
2. Report F-stat on the excluded instruments. F-stat < 10 means you have a weak instrument (Stock et al., 2002).
3. Present R^2 before and after including the instrument (Rossi, 2014)
4. For models with multiple instrument, present firs-t and second-stage result for each instrument separately. Overid test should be conducted (e.g., Sargan-Hansen J)
5. Hausman test between OLS and 2SLS (don't confuse this test for evidence that endogeneity is irrelevant - under invalid IV, the test is useless)
6. Compare the 2SLS with the limited information ML. If they are different, you have evidence for weak instruments.

36.1.3.2 Good Instruments

Exogeneity and Relevancy are necessary but not sufficient for IV to produce consistent estimates.

Without theory or possible explanation, you can always create a new variable that is correlated with X and uncorrelated with ϵ

For example, we want to estimate the effect of price on quantity (Reiss, 2011,

p. 960)

$$Q = \beta_1 P + \beta_2 X + \epsilon$$
$$P = \pi_1 X + \eta$$

where ϵ and η are jointly determined, $X \perp \epsilon, \eta$

Without theory, we can just create a new variable $Z = X + u$ where $E(u) = 0; u \perp X, \epsilon, \eta$

Then, Z satisfied both conditions:

- Relevancy: X correlates $P \rightarrow Z$ correlates P
- Exogeneity: $u \perp \epsilon$ (random noise)

But obviously, it's not a valid instrument (intuitively). But theoretically, relevance and exogeneity are not sufficient to identify β because of unsatisfied rank condition for identification.

Moreover, the functional form of the instrument also plays a role when choosing a good instrument. Hence, we always need to check for the robustness of our instrument.

IV methods even with valid instruments can still have poor sampling properties (finite sample bias, large sampling errors) (Rossi, 2014)

When you have a weak instrument, it's important to report it appropriately. This problem will be exacerbated if you have multiple instruments (Larcker and Rusticus, 2010).

36.1.3.2.1 Lagged dependent variable In time series data sets, we can use lagged dependent variable as an instrument because it is not influenced by current shocks. For example, Chetty et al. (2014) used lagged dependent variable in econ.

36.1.3.2.2 Lagged explanatory variable

- Common practice in applied economics: Replace suspected simultaneously determined explanatory variable with its lagged value (Reed, 2015, Bellemare et al. (2017)).
  - This practice does not avoid simultaneity bias.
  - Estimates using this method are still inconsistent.
  - Hypothesis testing becomes invalid under this approach.
  - Lagging variables changes how endogeneity bias operates, adding a "no dynamics among unobservables" assumption to the "selection on observables" assumption.

- Key conditions for appropriate use (Bellemare et al., 2017):
  - **Under unobserved confounding:**
    - No dynamics among unobservables.
    - The lagged variable X is a stationary autoregressive process.
  - **Under no unobserved confounding:**
    - No reverse causality; the causal effect operates with a one-period lag ($X_{t-1} \to Y$, $X_t \not\to Y_t$).
    - Reverse causality is contemporaneous, with a one-period lag effect.
    - Reverse causality is contemporaneous; no dynamics in Y, but dynamics exist in X ($X_{t-1} \to X$).
- **Alternative approach**: Use lagged values of the endogenous variable in IV estimation. However, IV estimation is only effective if (Reed, 2015):
  - Lagged values do not belong in the estimating equation.
  - Lagged values are sufficiently correlated with the simultaneously determined explanatory variable.
  - Lagged IVs help mitigate endogeneity if they only violate the independence assumption. However, if lagged IVs violate both the independence assumption and exclusion restriction, they may aggravate endogeneity (Wang and Bellemare, 2019).

36.1.3.3 Internal instrumental variable

- (also known as **instrument free methods**). This section is based on Raluca Gui's guide
- alternative to external instrumental variable approaches
- All approaches here assume a **continuous dependent variable**

36.1.3.3.1 Non-hierarchical Data (Cross-classified)

$$Y_t = \beta_0 + \beta_1 P_t + \beta_2 X_t + \epsilon_t$$

where

- $t = 1, .., T$ (indexes either time or cross-sectional units)
- Y_t is a $k \times 1$ response variable
- X_t is a $k \times n$ exogenous regressor
- P_t is a $k \times 1$ continuous endogenous regressor
- ϵ_t is a structural error term with $\mu_\epsilon = 0$ and $E(\epsilon^2) = \sigma^2$
- β are model parameters

The endogeneity problem arises from the correlation of P_t and ϵ_t:

$$P_t = \gamma Z_t + v_t$$

where

- Z_t is a $l \times 1$ vector of internal instrumental variables
- ν_t is a random error with $\mu_{v_t}, E(v^2) = \sigma_v^2, E(\epsilon v) = \sigma_{\epsilon v}$
- Z_t is assumed to be stochastic with distribution G
- ν_t is assumed to have density $h(\cdot)$

36.1.3.3.1.1 Latent Instrumental Variable (Ebbes et al., 2005)

assume Z_t (unobserved) to be uncorrelated with ϵ_t, which is similar to Instrumental Variable. Hence, Z_t and ν_t can't be identified without distributional assumptions

The distributions of Z_t and ν_t need to be specified such that:

(1) endogeneity of P_t is corrected
(2) the distribution of P_t is empirically close to the integral that expresses the amount of overlap of Z as it is shifted over (= the convolution between Z_t and ν_t).

When the density h(·) = Normal, then G cannot be normal because the parameters would not be identified (Ebbes et al., 2005) .

Hence,

- in the LIV model the distribution of Z_t is discrete
- in the Higher Moments Method and Joint Estimation Using Copula methods, the distribution of Z_t is taken to be skewed.

Z_t are assumed **unobserved, discrete and exogenous**, with

- an unknown number of groups m
- γ is a vector of group means.

Identification of the parameters relies on the distributional assumptions of

- P_t: a non-Gaussian distribution
- Z_t discrete with $m \geq 2$

Note:

- If Z_t is continuous, the model is unidentified
- If $P_t \sim N$, you have inefficient estimates.

```
m3.liv <- latentIV(read ~ stratio, data = school)
summary(m3.liv)$coefficients[1:7, ]
#>                  Estimate    Std. Error       z-score     Pr(>|z|)
#> (Intercept)   6.996014e+02  2.686165e+02  2.604462e+00 9.529035e-03
#> stratio      -2.272673e+00  1.367747e+01 -1.661618e-01 8.681097e-01
```

```
#> pi1          -4.896363e+01            NaN            NaN           NaN
#> pi2           1.963920e+01  9.225351e-02  2.128830e+02 0.000000e+00
#> theta5       6.939432e-152 3.143456e-161  2.207581e+09 0.000000e+00
#> theta6        3.787512e+02  4.249436e+01  8.912976e+00 1.541010e-17
#> theta7       -1.227543e+00  4.885237e+01 -2.512761e-02 9.799651e-01
```

it will return a coefficient very different from the other methods since there is only one endogenous variable.

36.1.3.3.1.2 Joint Estimation Using Copula assume Z_t (unobserved) to be uncorrelated with ϵ_t, which is similar to Instrumental Variable. Hence, Z_t and ν_t can't be identified without distributional assumptions

(Park and Gupta, 2012) allows joint estimation of the continuous P_t and ϵ_t using Gaussian copulas, where a copula is a function that maps several conditional distribution functions (CDF) into their joint CDF).

The underlying idea is that using information contained in the observed data, one selects marginal distributions for P_t and ϵ_t. Then, the copula model constructs a flexible multivariate joint distribution that allows a wide range of correlations between the two marginals.

The method allows both continuous and discrete P_t.

In the special case of **one continuous** P_t, estimation is based on MLE
Otherwise, based on Gaussian copulas, augmented OLS estimation is used.

Assumptions:

- skewed P_t
- the recovery of the correct parameter estimates
- $\epsilon_t \sim$ normal marginal distribution. The marginal distribution of P_t is obtained using the **Epanechnikov kernel density estimator**

$$\hat{h}_p = \frac{1}{T.b} \sum_{t=1}^{T} K(\frac{p - P_t}{b})$$

  where

- P_t = endogenous variables
- $K(x) = 0.75(1 - x^2)I(||x|| \leq 1)$
- $b = 0.9T^{-1/5} \times min(s, IQR/1.34)$
  - IQR = interquartile range
  - s = sample standard deviation
  - T = n of time periods observed in the data

```
# 1.34 comes from this
diff(qnorm(c(0.25, 0.75)))
#> [1] 1.34898
```

In augmented OLS and MLE, the inference procedure occurs in two stages:

(1): the empirical distribution of P_t is computed
(2) used in it constructing the likelihood function)
Hence, the standard errors would not be correct.

So we use the sampling distributions (from bootstrapping) to get standard errors and the variance-covariance matrix. Since the distribution of the bootstrapped parameters is highly skewed, we report the percentile confidence intervals is preferable.

```
set.seed(110)
m4.cc <-
    copulaCorrection(
        read ~ stratio + english + lunch + calworks +
            grades + income + county |
            continuous(stratio),
        data = school,
        optimx.args = list(method = c("Nelder-Mead"),
                           itnmax = 60000),
        num.boots = 2,
        verbose = FALSE
    )
summary(m4.cc)$coefficients[1:7,]
#>             Point Estimate   Boots SE Lower Boots CI (95%) Upper Boots CI (95%)
#> (Intercept)   682.25380724 2.80554213                   NA                    NA
#> stratio        -0.35704030 0.02075999                   NA                    NA
#> english        -0.21753937 0.01450666                   NA                    NA
#> lunch          -0.35642639 0.01902052                   NA                    NA
#> calworks       -0.06930202 0.02076781                   NA                    NA
#> gradesKK-08    -2.02155911 0.25684614                   NA                    NA
#> income          0.80137171 0.04725700                   NA                    NA
```

we run this model with only one endogenous continuous regressor (`stratio`). Sometimes, the code will not converge, in which case you can use different

- optimization algorithm
- starting values
- maximum number of iterations

36.1.3.3.1.3 Higher Moments Method suggested by (Lewbel, 1997) to identify ϵ_t caused by **measurement error**.

Identification is achieved by using third moments of the data, with no restric-

tions on the distribution of ϵ_t
The following instruments can be used with 2SLS estimation to obtain consistent estimates:

$$
\begin{aligned}
q_{1t} &= (G_t - \bar{G}) \\
q_{2t} &= (G_t - \bar{G})(P_t - \bar{P}) \\
q_{3t} &= (G_t - \bar{G})(Y_t - \bar{Y}) \\
q_{4t} &= (Y_t - \bar{Y})(P_t - \bar{P}) \\
q_{5t} &= (P_t - \bar{P})^2 \\
q_{6t} &= (Y_t - \bar{Y})^2
\end{aligned}
$$

where

- $G_t = G(X_t)$ for any given function G that has finite third own and cross moments
- $X =$ exogenous variable

q_{5t}, q_{6t} can be used only when the measurement and ϵ_t are symmetrically distributed. The rest of the instruments does not require any distributional assumptions for ϵ_t.

Since the regressors $G(X) = X$ are included as instruments, $G(X)$ can't be a linear function of X in q_{1t}

Since this method has very strong assumptions, Higher Moments Method should only be used in case of overidentification

```
set.seed(111)
m5.hetEr <-
    hetErrorsIV(
        read ~ stratio + english + lunch + calworks + income +
            grades + county |
            stratio | IIV(income, english),
        data = school
    )
summary(m5.hetEr)$coefficients[1:7,]
#>                 Estimate  Std. Error    t value     Pr(>|t|)
#> (Intercept) 662.78791557 27.90173069 23.7543657 2.380436e-76
#> stratio       0.71480686  1.31077325  0.5453322 5.858545e-01
#> english      -0.19522271  0.04057527 -4.8113717 2.188618e-06
#> lunch        -0.37834232  0.03927793 -9.6324402 9.760809e-20
#> calworks     -0.05665126  0.06302095 -0.8989273 3.692776e-01
#> income        0.82693755  0.17236557  4.7975797 2.335271e-06
#> gradesKK-08  -1.93795843  1.38723186 -1.3969968 1.632541e-01
```

recommend using this approach to create additional instruments to use with external ones for better efficiency.

36.1.3.3.1.4 Heteroskedastic Error Approach

- using means of variables that are uncorrelated with the product of heteroskedastic errors to identify structural parameters.
- This method can be use either when you don't have external instruments or you want to use additional instruments to improve the efficiency of the IV estimator (Lewbel, 2012)
- The instruments are constructed as simple functions of data
- Model's assumptions:

$$\begin{aligned} E(X\epsilon) &= 0 \\ E(Xv) &= 0 \\ cov(Z, \epsilon v) &= 0 \\ cov(Z, v^2) &\neq 0 \text{ (for identification)} \end{aligned}$$

Structural parameters are identified by 2SLS regression of Y on X and P, using X and [Z − E(Z)] as instruments.

$$\text{instrument's strength} \propto cov((Z - \bar{Z})v, v)$$

where $cov((Z - \bar{Z})v, v)$ is the degree of heteroskedasticity of with respect to Z (Lewbel, 2012), which can be empirically tested.

If it is zero or close to zero (i.e.,the instrument is weak), you might have imprecise estimates, with large standard errors.

- Under homoskedasticity, the parameters of the model are unidentified.
- Under heteroskedasticity related to at least some elements of X, the parameters of the model are identified.

36.1.3.3.2 Hierarchical Data Multiple independent assumptions involving various random components at different levels mean that any moderate correlation between some predictors and a random component or error term can result in a significant bias of the coefficients and of the variance components. (Kim and Frees, 2007) proposed a generalized method of moments which uses both, the between and within variations of the exogenous variables, but only assumes the within variation of the variables to be endogenous.

Assumptions

- the errors at each level $\sim iidN$
- the slope variables are exogenous
- the level-1 $\epsilon \perp X, P$. If this is not the case, additional, external instruments are necessary

Hierarchical Model

$$
\begin{aligned}
Y_{cst} &= Z^1_{cst}\beta^1_{cs} + X^1_{cst}\beta_1 + \epsilon^1_{cst} \\
\beta^1_{cs} &= Z^2_{cs}\beta^2_c + X^2_{cst}\beta_2 + \epsilon^2_{cst} \\
\beta^2_c &= X^3_c\beta_3 + \epsilon^3_c
\end{aligned}
$$

Bias could stem from:

- errors at the higher two levels ($\epsilon^3_c, \epsilon^2_{cst}$) are correlated with some of the regressors
- only third level errors (ϵ^3_c) are correlated with some of the regressors

(Kim and Frees, 2007) proposed

- When all variables are assumed exogenous, the proposed estimator equals the random effects estimator
- When all variables are assumed endogenous, it equals the fixed effects estimator
- also use omitted variable test (based on the Hausman-test (Hausman, 1978) for panel data), which allows the comparison of a robust estimator and an estimator that is efficient under the null hypothesis of no omitted variables or the comparison of two robust estimators at different levels.

```
# function 'cholmod_factor_ldetA' not provided by package 'Matrix'
set.seed(113)
school$gr08 <- school$grades == "KK-06"
m7.multilevel <-
    multilevelIV(read ~ stratio + english + lunch + income + gr08 +
                     calworks + (1 | county) | endo(stratio),
                 data = school)
summary(m7.multilevel)$coefficients[1:7,]
```

Another example using simulated data

- level-1 regressors: $X_{11}, X_{12}, X_{13}, X_{14}, X_{15}$, where X_{15} is correlated with the level-2 error (i.e., endogenous).

- level-2 regressors: $X_{21}, X_{22}, X_{23}, X_{24}$

- level-3 regressors: X_{31}, X_{32}, X_{33}

We estimate a three-level model with X15 assumed endogenous. Having a three-level hierarchy, `multilevelIV()` returns five estimators, from the most robust to omitted variables (FE_L2), to the most efficient (REF) (i.e. lowest mean squared error).

- The random effects estimator (REF) is efficient assuming no omitted variables

- The fixed effects estimator (FE) is unbiased and asymptotically normal even in the presence of omitted variables.
- Because of the efficiency, the random effects estimator is preferable if you think there is no omitted. variables
- The robust estimator would be preferable if you think there is omitted variables.

```
# function 'cholmod_factor_ldetA' not provided by package 'Matrix''
data(dataMultilevelIV)
set.seed(114)
formula1 <-
    y ~ X11 + X12 + X13 + X14 + X15 + X21 + X22 + X23 + X24 +
    X31 + X32 + X33 + (1 | CID) + (1 | SID) | endo(X15)
m8.multilevel <-
    multilevelIV(formula = formula1, data = dataMultilevelIV)
coef(m8.multilevel)

summary(m8.multilevel, "REF")
```

True $\beta_{X_{15}} = -1$. We can see that some estimators are bias because X_{15} is correlated with the level-two error, to which only FE_L2 and GMM_L2 are robust

To select the appropriate estimator, we use the omitted variable test.

In a three-level setting, we can have different estimator comparisons:

- Fixed effects vs. random effects estimators: Test for omitted level-two and level-three omitted effects, simultaneously, one compares FE_L2 to REF. But we will not know at which omitted variables exist.

- Fixed effects vs. GMM estimators: Once the existence of omitted effects is established but not sure at which level, we test for level-2 omitted effects by comparing FE_L2 vs GMM_L3. If you reject the null, the omitted variables are at level-2 The same is accomplished by testing FE_L2 vs. GMM_L2, since the latter is consistent only if there are no omitted effects at level-2.

- Fixed effects vs. fixed effects estimators: We can test for omitted level-2 effects, while allowing for omitted level-3 effects by comparing FE_L2 vs. FE_L3 since FE_L2 is robust against both level-2 and level-3 omitted effects while FE_L3 is only robust to level-3 omitted variables.

Summary, use the omitted variable test comparing `REF vs. FE_L2` first.

- If the null hypothesis is rejected, then there are omitted variables either at level-2 or level-3

- Next, test whether there are level-2 omitted effects, since testing for omit-

ted level three effects relies on the assumption there are no level-two omitted effects. You can use any of these pair of comparisons:

  - `FE_L2 vs. FE_L3`
  - `FE_L2 vs. GMM_L2`

- If no omitted variables at level-2 are found, test for omitted level-3 effects by comparing either

  - `FE_L3` vs. `GMM_L3`
  - `GMM_L2` vs. `GMM_L3`

```
summary(m8.multilevel, "REF")
# compare REF with all the other estimators. Testing REF (the most efficient estimator) against F
```

Since the null hypothesis is rejected (p = 0.000139), there is bias in the random effects estimator.

To test for level-2 omitted effects (regardless of level-3 omitted effects), we compare FE_L2 versus FE_L3

```
summary(m8.multilevel,"FE_L2")
```

The null hypothesis of no omitted level-2 effects is rejected ($p = 3.92e - 05$). Hence, there are omitted effects at level-two. We should use FE_L2 which is consistent with the underlying data that we generated (level-2 error correlated with $X_1 5$, which leads to biased FE_L3 coefficients.

The omitted variable test between FE_L2 and GMM_L2 should reject the null hypothesis of no omitted level-2 effects (p-value is 0).

If we assume an endogenous variable as exogenous, the RE and GMM estimators will be biased because of the wrong set of internal instrumental variables. To increase our confidence, we should compare the omitted variable tests when the variable is considered endogenous vs. exogenous to get a sense whether the variable is truly endogenous.

36.1.3.4 Proxy Variables

- Can be in place of the omitted variable
- will not be able to estimate the effect of the omitted variable
- will be able to reduce some endogeneity caused bye the omitted variable
- but it can have Measurement Error. Hence, you have to be extremely careful when using proxies.

Criteria for a proxy variable:

1. The proxy is correlated with the omitted variable.
2. Having the omitted variable in the regression will solve the problem of endogeneity

3. The variation of the omitted variable unexplained by the proxy is uncorrelated with all independent variables, including the proxy.

IQ test can be a proxy for ability in the regression between wage explained education.

For the third requirement

$$ability = \gamma_0 + \gamma_1 IQ + \epsilon$$

where ϵ is uncorrelated with education and IQ test.

36.2 Endogenous Sample Selection

Selection into treatment does not occur randomly in observational studies, leading to **selection bias**—a major challenge in causal inference. Individuals often choose whether to participate in a treatment based on personal characteristics, external incentives, or underlying risk factors. This selection process can introduce systematic differences between the treatment and control groups, biasing the estimated treatment effects.

Selection bias typically arises from two opposing sources:

36.2.1 Mitigation-Based Selection

- Individuals **select into treatment to combat a problem** they already face.
- This creates a **negative selection bias**—those who take treatment are systematically worse off compared to those who do not.
- **Example**:
  - People at **high risk of severe illness** (e.g., elderly or immunocompromised individuals) are more likely to **get vaccinated**. If we compare vaccinated vs. unvaccinated individuals without adjusting for risk factors, we might mistakenly conclude that vaccines are ineffective simply because vaccinated individuals had worse initial health conditions.

36.2.2 Preference-Based Selection

- Individuals **select into treatment because they inherently prefer it**, rather than because of an underlying problem.
- This creates a **positive selection bias**—those who take treatment are systematically better off compared to those who do not.
- **Example**:
  - People who are **health-conscious and physically active** are more likely to **join a fitness program**. If we compare fitness program participants to non-participants, we might falsely attribute their better

health outcomes to the program, when in reality, their pre-existing lifestyle contributed to their improved health.

36.3 Implications for Causal Inference

Selection bias distorts the observed treatment effect, leading to **biased estimates of causal relationships**. Depending on the nature of selection, this bias can **overestimate or underestimate** the true effect:

- **Negative Selection Bias (Mitigation-Based)**:
  - The observed treatment group appears **worse off** than the control group.
  - The treatment effect may be **underestimated** or even appear harmful.
  - Example: Evaluating the effect of **job training programs**—unemployed individuals with the greatest difficulty finding jobs are most likely to enroll, leading to underestimated program benefits.
- **Positive Selection Bias (Preference-Based)**:
  - The observed treatment group appears **better off** than the control group.
  - The treatment effect may be **overestimated**.
  - Example: Evaluating the effect of **private school education**—students who attend private schools often come from wealthier families with greater academic support, making it difficult to isolate the true impact of the school itself.

36.3.1 Addressing Selection Bias

Several causal inference methods aim to **correct for selection bias** and estimate unbiased treatment effects:

1. **Randomized Controlled Trials (RCTs)**
   - Gold standard for causal inference, eliminating selection bias by randomly assigning treatment.
2. **Instrumental Variables (IV)**
   - Uses an external variable (instrument) that affects treatment assignment but is unrelated to the outcome, helping to isolate the causal effect.
3. **Difference-in-Differences (DiD)**
   - Compares changes in outcomes before and after treatment between a treatment and control group, assuming parallel trends.
4. **Matching Methods (e.g., Propensity Score Matching)**
   - Creates a synthetic control group by matching treated and untreated individuals with similar observable characteristics.
5. **Regression Discontinuity (RD)**

    - Exploits treatment assignment thresholds (e.g., age cutoffs for eligibility) to compare individuals just above and below the threshold.
6. **Synthetic Control Methods**
    - Constructs a weighted combination of control units to approximate the counterfactual outcome of the treated group.

- Also known as sample selection or self-selection problem or incidental truncation.

- The omitted variable is how people were selected into the sample

Some disciplines consider nonresponse bias and selection bias as sample selection.

- When unobservable factors that affect who is in the sample are independent of unobservable factors that affect the outcome, the sample selection is not endogenous. Hence, the sample selection is ignorable and estimator that ignores sample selection is still consistent.
- when the unobservable factors that affect who is included in the sample are correlated with the unobservable factors that affect the outcome, the sample selection is endogenous and not ignorable, because estimators that ignore endogenous sample selection are not consistent (we don't know which part of the observable outcome is related to the causal relationship and which part is due to different people were selected for the treatment and control groups).

Assumptions: - The unobservables that affect the treatment selection and the outcome are jointly distributed as bivariate normal.

Notes:

- If you don't have strong exclusion restriction, identification is driven by the assumed non linearity in the functional form (through inverse Mills ratio). E.g., the estimate depend on the bivariate normal distribution of the error structure:

    - With strong exclusion restriction for the covariate in the correction equation, the variation in this variable can help identify the control for selection
    - With weak exclusion restriction, and the variable exists in both steps, it's the assumed error structure that identifies the control for selection (Heckman and Navarro-Lozano, 2004).

- In management, Wolfolds and Siegel (2019) found that papers should have valid exclusion conditions, because without these, simulations show that results using the Heckman method are less reliable than those obtained with OLS.

- There are differences between Heckman Sample Selection vs. Heckman-type correction

	Heckman Sample Selection Model	**Heckman-Type Corrections**
When	Only observes one sample (treated), addressing selection bias directly.	Two samples are observed (treated and untreated), known as the control function approach.
Model	Probit	OLS (even for dummy endogenous variable)
Integration of 1st stage	Also include a term (called Inverse Mills ratio) besides the endogenous variable.	Decompose the endogenous variable to get the part that is uncorrelated with the error terms of the outcome equation. Either use the predicted endogenous variable directly or include the residual from the first-stage equation.
Advantages and Assumptions	Provides a direct test for endogeneity via the coefficient of the inverse Mills ratio but requires the assumption of joint normality of errors.	Does not require the assumption of joint normality, but can't test for endogeneity directly.

To deal with [Sample Selection], we can

- Randomization: participants are randomly selected into treatment and control.
- Instruments that determine the treatment status (i.e., treatment vs. control) but not the outcome (Y)
- Functional form of the selection and outcome processes: originated from (Heckman, 1976), later on generalize by (Amemiya, 1984)

We have our main model

$$\mathbf{y}^* = \mathbf{xb} +$$

However, the pattern of missingness (i.e., censored) is related to the unobserved (latent) process:

$$\mathbf{z}^* = \mathbf{w} \ + \mathbf{u}$$

and

$$z_i = \begin{cases} 1 & \text{if } z_i^* > 0 \\ 0 & \text{if } z_i^* \leq 0 \end{cases}$$

Equivalently, $z_i = 1$ (y_i is observed) when

$$u_i \geq -w_i\gamma$$

Hence, the probability of observed y_i is

$$\begin{aligned} P(u_i \geq -w_i\gamma) &= 1 - \Phi(-w_i\gamma) \\ &= \Phi(w_i\gamma) \qquad \text{symmetry of the standard normal distribution} \end{aligned}$$

We will **assume**

- the error term of the selection $\mathbf{u} \sim \mathbf{N}(\mathbf{0}, \mathbf{I})$
- $Var(u_i) = 1$ for identification purposes

Visually, $P(u_i \geq -w_i\gamma)$ is the shaded area.

```
x = seq(-3, 3, length = 200)
y = dnorm(x, mean = 0, sd = 1)
plot(x,
     y,
     type = "l",
     main = bquote("Probabibility distribution of" ~ u[i]))
x = seq(0.3, 3, length = 100)
y = dnorm(x, mean = 0, sd = 1)
polygon(c(0.3, x, 3), c(0, y, 0), col = "gray")
text(1, 0.1, bquote(1 - Phi ~ (-w[i] ~ gamma)))
arrows(-0.5, 0.1, 0.3, 0, length = .15)
text(-0.5, 0.12, bquote(-w[i] ~ gamma))
legend(
    "topright",
    "Gray = Prob of Observed",
    pch = 1,
    title = "legend",
    inset = .02
)
```

Hence in our observed model, we see

$$y_i = x_i\beta + \epsilon_i \text{when } z_i = 1 \tag{36.2}$$

and the joint distribution of the selection model (u_i), and the observed equation (ϵ_i) as

$$\begin{bmatrix} u \\ \epsilon \end{bmatrix} \sim^{iid} N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho \\ \rho & \sigma_\epsilon^2 \end{bmatrix}\right)$$

The relation between the observed and selection models:

$$\begin{aligned} E(y_i|y_i \text{ observed}) &= E(y_i|z^* > 0) \\ &= E(y_i| - w_i\gamma) \\ &= \mathbf{x}_i\beta + E(\epsilon_i|u_i > -w_i\gamma) \\ &= \mathbf{x}_i\beta + \rho\sigma_\epsilon\frac{\phi(w_i\gamma)}{\Phi(w_i\gamma)} \end{aligned}$$

where $\frac{\phi(w_i\gamma)}{\Phi(w_i\gamma)}$ is the Inverse Mills Ratio. and $\rho\sigma_\epsilon\frac{\phi(w_i\gamma)}{\Phi(w_i\gamma)} \geq 0$

A property of IMR: Its derivative is: $IMR'(x) = -xIMR(x) - IMR(x)^2$

Great visualization of special cases of correlation patterns among data and errors by professor Rob Hick

Note:

(Bareinboim and Pearl, 2014) is an excellent summary of cases that we can still do causal inference in case of selection bias. I'll try to summarize their idea here:

Let X be an action, Y be an outcome, and S be a binary indicator of entry into the data pool where ($S = 1 =$ in the sample, $S = 0 =$ out of sample) and Q be the conditional distribution $Q = P(y|x)$.

Usually we want to understand , but because of S, we only have $P(y, x|S = 1)$. Hence, we'd like to recover $P(y|x)$ from $P(y, x|S = 1)$

- If both X and Y affect S, we can't unbiasedly estimate $P(y|x)$

In the case of Omitted variable bias (U) and sample selection bias (S), you have unblocked extraneous "flow" of information between X and Y, which causes spurious correlation for X and Y. Traditionally, we would recover Q by parametric assumption of

(1) The data generating process (e.g., Heckman 2-step)
(2) Type of data-generating model (e..g, treatment-dependent or outcome-dependent)
(3) Selection's probability $P(S = 1|Pa_s)$ with non-parametrically based causal graphical models, the authors proposed more robust way to model misspecification regardless of the type of data-generating model, and do not require selection's probability. Hence, you can recover Q
    - Without external data
    - With external data
    - Causal effects with the Selection-backdoor criterion

36.3.2 Tobit-2

also known as Heckman's standard sample selection model
Assumption: joint normality of the errors

Data here is taken from Mroz (1984).

We want to estimate the log(wage) for married women, with education, experience, experience squared, and a dummy variable for living in a big city. But we can only observe the wage for women who are working, which means a lot of married women in 1975 who were out of the labor force are unaccounted for. Hence, an OLS estimate of the wage equation would be bias due to sample selection. Since we have data on non-participants (i.e., those who are not working for pay), we can correct for the selection process.

The Tobit-2 estimates are consistent

36.3.2.1 Example 1

```
library(sampleSelection)
library(dplyr)
# 1975 data on married women's pay and labor-force participation
# from the Panel Study of Income Dynamics (PSID)
data("Mroz87")
head(Mroz87)
#>   lfp hours kids5 kids618 age educ   wage repwage hushrs husage huseduc huswage
#> 1   1  1610     1       0  32   12 3.3540    2.65   2708     34      12  4.0288
#> 2   1  1656     0       2  30   12 1.3889    2.65   2310     30       9  8.4416
#> 3   1  1980     1       3  35   12 4.5455    4.04   3072     40      12  3.5807
#> 4   1   456     0       3  34   12 1.0965    3.25   1920     53      10  3.5417
#> 5   1  1568     1       2  31   14 4.5918    3.60   2000     32      12 10.0000
#> 6   1  2032     0       0  54   12 4.7421    4.70   1040     57      11  6.7106
#>   faminc    mtr motheduc fatheduc unem city exper  nwifeinc wifecoll huscoll
#> 1  16310 0.7215       12        7  5.0    0    14 10.910060    FALSE   FALSE
#> 2  21800 0.6615        7        7 11.0    1     5 19.499981    FALSE   FALSE
#> 3  21040 0.6915       12        7  5.0    0    15 12.039910    FALSE   FALSE
#> 4   7300 0.7815        7        7  5.0    0     6  6.799996    FALSE   FALSE
#> 5  27300 0.6215       12       14  9.5    1     7 20.100058     TRUE   FALSE
#> 6  19495 0.6915       14        7  7.5    1    33  9.859054    FALSE   FALSE
Mroz87 = Mroz87 %>%
    mutate(kids = kids5 + kids618)

library(nnet)
library(ggplot2)
library(reshape2)
```

2-stage Heckman's model:

(1) probit equation estimates the selection process (who is in the labor force?)
(2) the results from 1st stage are used to construct a variable that captures the selection effect in the wage equation. This correction variable is called the **inverse Mills ratio**.

```
# OLS: log wage regression on LF participants only
ols1 = lm(log(wage) ~ educ + exper + I(exper ^ 2) + city,
          data = subset(Mroz87, lfp == 1))
# Heckman's Two-step estimation with LFP selection equation
heck1 = heckit(
    selection = lfp ~ age + I(age ^ 2) + kids + huswage + educ,
    # the selection process, l
    # fp = 1 if the woman is participating in the labor force
    outcome = log(wage) ~ educ + exper + I(exper ^ 2) + city,
    data = Mroz87
)
```

```
summary(heck1$probit)
#> --------------------------------------------
#> Probit binary choice model/Maximum Likelihood estimation
#> Newton-Raphson maximisation, 4 iterations
#> Return code 1: gradient close to zero (gradtol)
#> Log-Likelihood: -482.8212
#> Model: Y == '1' in contrary to '0'
#> 753 observations (325 'negative' and 428 'positive') and 6 free parameters (df = 74
#> Estimates:
#>                  Estimate  Std. error t value   Pr(> t)
#> XS(Intercept) -4.18146681  1.40241567 -2.9816  0.002867 **
#> XSage          0.18608901  0.06517476  2.8552  0.004301 **
#> XSI(age^2)    -0.00241491  0.00075857 -3.1835  0.001455 **
#> XSkids        -0.14955977  0.03825079 -3.9100 9.230e-05 ***
#> XShuswage     -0.04303635  0.01220791 -3.5253  0.000423 ***
#> XSeduc         0.12502818  0.02277645  5.4894 4.034e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> Significance test:
#> chi2(5) = 64.10407 (p=1.719042e-12)
#> --------------------------------------------
summary(heck1$lm)
#>
#> Call:
#> lm(formula = YO ~ -1 + XO + imrData$IMR1, subset = YS == 1, weights = weightsNoNA)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -3.09494 -0.30953  0.05341  0.36530  2.34770
#>
#> Coefficients:
#>                 Estimate Std. Error t value Pr(>|t|)
#> XO(Intercept) -0.6143381  0.3768796  -1.630  0.10383
#> XOeduc         0.1092363  0.0197062   5.543 5.24e-08 ***
#> XOexper        0.0419205  0.0136176   3.078  0.00222 **
#> XOI(exper^2)  -0.0008226  0.0004059  -2.026  0.04335 *
#> XOcity         0.0510492  0.0692414   0.737  0.46137
#> imrData$IMR1   0.0551177  0.2111916   0.261  0.79423
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.6674 on 422 degrees of freedom
#> Multiple R-squared:  0.7734, Adjusted R-squared:  0.7702
#> F-statistic:   240 on 6 and 422 DF,  p-value: < 2.2e-16
```

Use only variables that affect the selection process in the selection equation. Technically, the selection equation and the equation of interest could have the same set of regressors. But it is not recommended because we should only use variables (or at least one) in the selection equation that affect the selection process, but not the wage process (i.e., instruments). Here, variable `kids` fulfill that role: women with kids may be more likely to stay home, but working moms with kids would not have their wages change.

Alternatively,

```
# ML estimation of selection model
ml1 = selection(
    selection = lfp ~ age + I(age ^ 2) + kids + huswage + educ,
    outcome = log(wage) ~ educ + exper + I(exper ^ 2) + city,
    data = Mroz87
)
summary(ml1)
#> --------------------------------------------
#> Tobit 2 model (sample selection model)
#> Maximum Likelihood estimation
#> Newton-Raphson maximisation, 3 iterations
#> Return code 8: successive function values within relative tolerance limit (reltol)
#> Log-Likelihood: -914.0777
#> 753 observations (325 censored and 428 observed)
#> 13 free parameters (df = 740)
#> Probit selection equation:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -4.1484037  1.4109302  -2.940 0.003382 **
#> age          0.1842132  0.0658041   2.799 0.005253 **
#> I(age^2)    -0.0023925  0.0007664  -3.122 0.001868 **
#> kids        -0.1488158  0.0384888  -3.866 0.000120 ***
#> huswage     -0.0434253  0.0123229  -3.524 0.000451 ***
#> educ         0.1255639  0.0229229   5.478 5.91e-08 ***
#> Outcome equation:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.5814781  0.3052031  -1.905  0.05714 .
#> educ         0.1078481  0.0172998   6.234 7.63e-10 ***
#> exper        0.0415752  0.0133269   3.120  0.00188 **
#> I(exper^2)  -0.0008125  0.0003974  -2.044  0.04129 *
#> city         0.0522990  0.0682652   0.766  0.44385
#>    Error terms:
#>       Estimate Std. Error t value Pr(>|t|)
#> sigma  0.66326    0.02309  28.729   <2e-16 ***
#> rho    0.05048    0.23169   0.218    0.828
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#> ---------------------------------------------
# summary(ml1$twoStep)
```

Manual

```
myprob <- probit(lfp ~ age + I( age^2 ) + kids + huswage + educ,
                 # x = TRUE,
                 # iterlim = 30,
                 data = Mroz87)
summary(myprob)
#> ---------------------------------------------
#> Probit binary choice model/Maximum Likelihood estimation
#> Newton-Raphson maximisation, 4 iterations
#> Return code 1: gradient close to zero (gradtol)
#> Log-Likelihood: -482.8212
#> Model: Y == '1' in contrary to '0'
#> 753 observations (325 'negative' and 428 'positive') and 6 free parameters (df = 74
#> Estimates:
#>                Estimate  Std. error t value   Pr(> t)
#> (Intercept) -4.18146681  1.40241567 -2.9816  0.002867 **
#> age          0.18608901  0.06517476  2.8552  0.004301 **
#> I(age^2)    -0.00241491  0.00075857 -3.1835  0.001455 **
#> kids        -0.14955977  0.03825079 -3.9100 9.230e-05 ***
#> huswage     -0.04303635  0.01220791 -3.5253  0.000423 ***
#> educ         0.12502818  0.02277645  5.4894 4.034e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> Significance test:
#> chi2(5) = 64.10407 (p=1.719042e-12)
#> ---------------------------------------------

imr <- invMillsRatio(myprob)
Mroz87$IMR1 <- imr$IMR1

manually_est <- lm(log(wage) ~ educ + exper + I( exper^2 ) + city + IMR1,
                   data = Mroz87,
                   subset = (lfp == 1))

summary(manually_est)
#>
#> Call:
#> lm(formula = log(wage) ~ educ + exper + I(exper^2) + city + IMR1,
#>     data = Mroz87, subset = (lfp == 1))
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
```

```
#> -3.09494 -0.30953  0.05341  0.36530  2.34770
#>
#> Coefficients:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.6143381  0.3768796  -1.630  0.10383
#> educ         0.1092363  0.0197062   5.543 5.24e-08 ***
#> exper        0.0419205  0.0136176   3.078  0.00222 **
#> I(exper^2)  -0.0008226  0.0004059  -2.026  0.04335 *
#> city         0.0510492  0.0692414   0.737  0.46137
#> IMR1         0.0551177  0.2111916   0.261  0.79423
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.6674 on 422 degrees of freedom
#> Multiple R-squared:  0.1582, Adjusted R-squared:  0.1482
#> F-statistic: 15.86 on 5 and 422 DF,  p-value: 2.505e-14
```

Similarly,

```
probit_selection <-
    glm(lfp ~ age + I( age^2 ) + kids + huswage + educ,
        data = Mroz87,
        family = binomial(link = 'probit'))

# library(fixest)
# probit_selection <-
#     fixest::feglm(lfp ~ age + I( age^2 ) + kids + huswage + educ,
#         data = Mroz87,
#         family = binomial(link = 'probit'))

probit_lp <- -predict(probit_selection)
inv_mills <- dnorm(probit_lp) / (1 - pnorm(probit_lp))
Mroz87$inv_mills <- inv_mills


probit_outcome <-
    glm(
        log(wage) ~ educ + exper + I(exper ^ 2) + city + inv_mills,
        data = Mroz87,
        subset = (lfp == 1)
    )
summary(probit_outcome)
#>
#> Call:
#> glm(formula = log(wage) ~ educ + exper + I(exper^2) + city +
#>     inv_mills, data = Mroz87, subset = (lfp == 1))
```

```
#>
#> Coefficients:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.6143383  0.3768798  -1.630  0.10383
#> educ         0.1092363  0.0197062   5.543 5.24e-08 ***
#> exper        0.0419205  0.0136176   3.078  0.00222 **
#> I(exper^2)  -0.0008226  0.0004059  -2.026  0.04335 *
#> city         0.0510492  0.0692414   0.737  0.46137
#> inv_mills    0.0551179  0.2111918   0.261  0.79423
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 0.4454809)
#>
#>     Null deviance: 223.33  on 427  degrees of freedom
#> Residual deviance: 187.99  on 422  degrees of freedom
#> AIC: 876.49
#>
#> Number of Fisher Scoring iterations: 2
```

```r
library("stargazer")
library("Mediana")
library("plm")
# function to calculate corrected SEs for regression
cse = function(reg) {
  rob = sqrt(diag(vcovHC(reg, type = "HC1")))
  return(rob)
}

# stargazer table
stargazer(
    # ols1,
    heck1,
    ml1,
    # manually_est,

    se = list(cse(ols1), NULL, NULL),
    title = "Married women's wage regressions",
    type = "text",
    df = FALSE,
    digits = 4,
    selection.equation = T
)
#>
#> Married women's wage regressions
```

```
#> ====================================================
#>                           Dependent variable:
#>                       -------------------------------
#>                                   lfp
#>                        Heckman        selection
#>                       selection
#>                          (1)             (2)
#> ----------------------------------------------------
#> age                   0.1861***       0.1842***
#>                                       (0.0658)
#>
#> I(age2)                -0.0024       -0.0024***
#>                                       (0.0008)
#>
#> kids                 -0.1496***      -0.1488***
#>                                       (0.0385)
#>
#> huswage                -0.0430       -0.0434***
#>                                       (0.0123)
#>
#> educ                   0.1250         0.1256***
#>                       (0.0130)        (0.0229)
#>
#> Constant             -4.1815***      -4.1484***
#>                       (0.2032)        (1.4109)
#>
#> ----------------------------------------------------
#> Observations             753             753
#> R2                     0.1582
#> Adjusted R2            0.1482
#> Log Likelihood                        -914.0777
#> rho                    0.0830      0.0505 (0.2317)
#> Inverse Mills Ratio 0.0551 (0.2099)
#> ====================================================
#> Note:                  *p<0.1; **p<0.05; ***p<0.01
```

```r
stargazer(
    ols1,
    # heck1,
    # ml1,
    manually_est,

    se = list(cse(ols1), NULL, NULL),
    title = "Married women's wage regressions",
```

```
    type = "text",
    df = FALSE,
    digits = 4,
    selection.equation = T
)
#>
#> Married women's wage regressions
#> ==================================================
#>                          Dependent variable:
#>                      ----------------------------
#>                              log(wage)
#>                           (1)            (2)
#> --------------------------------------------------
#> educ                   0.1057***      0.1092***
#>                        (0.0130)       (0.0197)
#>
#> exper                  0.0411***      0.0419***
#>                        (0.0154)       (0.0136)
#>
#> I(exper2)              -0.0008*       -0.0008**
#>                        (0.0004)       (0.0004)
#>
#> city                    0.0542         0.0510
#>                        (0.0653)       (0.0692)
#>
#> IMR1                                   0.0551
#>                                       (0.2112)
#>
#> Constant               -0.5308***      -0.6143
#>                        (0.2032)       (0.3769)
#>
#> --------------------------------------------------
#> Observations              428            428
#> R2                       0.1581         0.1582
#> Adjusted R2              0.1501         0.1482
#> Residual Std. Error      0.6667         0.6674
#> F Statistic            19.8561***     15.8635***
#> ==================================================
#> Note:                  *p<0.1; **p<0.05; ***p<0.01
```

Rho is an estimate of the correlation of the errors between the selection and wage equations. In the lower panel, the estimated coefficient on the inverse Mills ratio is given for the Heckman model. The fact that it is not statistically different from zero is consistent with the idea that selection bias was not a serious problem in this case.

If the estimated coefficient of the inverse Mills ratio in the Heckman model is not statistically different from zero, then selection bias was not a serious problem.

36.3.2.2 Example 2

This code is from R package sampleSelection

```
set.seed(0)
library("sampleSelection")
library("mvtnorm")
# bivariate normal disturbances
eps <-
    rmvnorm(500, c(0, 0), matrix(c(1, -0.7, -0.7, 1), 2, 2))

# uniformly distributed explanatory variable
# (vectors of explanatory variables for the selection)
xs <- runif(500)

# probit data generating process
ys <- xs + eps[, 1] > 0

# vectors of explanatory variables for outcome equation
xo <- runif(500)
yoX <- xo + eps[, 2] # latent outcome
yo <- yoX * (ys > 0) # observable outcome
# true intercepts = 0 and our true slopes = 1
# xs and xo are independent.
# Hence, exclusion restriction is fulfilled
summary(selection(ys ~ xs, yo ~ xo))
#> --------------------------------------------
#> Tobit 2 model (sample selection model)
#> Maximum Likelihood estimation
#> Newton-Raphson maximisation, 5 iterations
#> Return code 1: gradient close to zero (gradtol)
#> Log-Likelihood: -712.3163
#> 500 observations (172 censored and 328 observed)
#> 6 free parameters (df = 494)
#> Probit selection equation:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  -0.2228     0.1081  -2.061   0.0399 *
#> xs            1.3377     0.2014   6.642 8.18e-11 ***
#> Outcome equation:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.0002265  0.1294178  -0.002    0.999
#> xo           0.7299070  0.1635925   4.462 1.01e-05 ***
#>    Error terms:
```

```
#>       Estimate Std. Error t value Pr(>|t|)
#> sigma   0.9190     0.0574  16.009  < 2e-16 ***
#> rho    -0.5392     0.1521  -3.544 0.000431 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> --------------------------------------------
```

without the exclusion restriction, we generate yo using xs instead of xo.

```
yoX <- xs + eps[,2]
yo <- yoX*(ys > 0)
summary(selection(ys ~ xs, yo ~ xs))
#> --------------------------------------------
#> Tobit 2 model (sample selection model)
#> Maximum Likelihood estimation
#> Newton-Raphson maximisation, 14 iterations
#> Return code 8: successive function values within relative tolerance limit (reltol)
#> Log-Likelihood: -712.8298
#> 500 observations (172 censored and 328 observed)
#> 6 free parameters (df = 494)
#> Probit selection equation:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  -0.1984     0.1114  -1.781   0.0756 .
#> xs            1.2907     0.2085   6.191 1.25e-09 ***
#> Outcome equation:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  -0.5499     0.5644  -0.974  0.33038
#> xs            1.3987     0.4482   3.120  0.00191 **
#>    Error terms:
#>       Estimate Std. Error t value Pr(>|t|)
#> sigma  0.85091    0.05352  15.899   <2e-16 ***
#> rho   -0.13226    0.72684  -0.182    0.856
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> --------------------------------------------
```

We can see that our estimates are still unbiased but standard errors are substantially larger. The exclusion restriction (i.e., independent information about the selection process) has a certain identifying power that we desire. Hence, it's better to have different set of variable for the selection process from the interested equation. Without the exclusion restriction, we solely rely on the functional form identification.

36.3.3 Tobit-5

Also known as the switching regression model
Condition: There is at least one variable in X in the selection process not

included in the observed process. Used when there are separate models for participants, and non-participants.

```
set.seed(0)
vc <- diag(3)
vc[lower.tri(vc)] <- c(0.9, 0.5, 0.1)
vc[upper.tri(vc)] <- vc[lower.tri(vc)]

# 3 disturbance vectors by a 3-dimensional normal distribution
eps <- rmvnorm(500, c(0,0,0), vc)
xs <- runif(500) # uniformly distributed on [0, 1]
ys <- xs + eps[,1] > 0
xo1 <- runif(500) # uniformly distributed on [0, 1]
yo1 <- xo1 + eps[,2]
xo2 <- runif(500) # uniformly distributed on [0, 1]
yo2 <- xo2 + eps[,3]
```

exclusion restriction is fulfilled when x's are independent.

```
# one selection equation and a list of two outcome equations
summary(selection(ys~xs, list(yo1 ~ xo1, yo2 ~ xo2)))
#> --------------------------------------------
#> Tobit 5 model (switching regression model)
#> Maximum Likelihood estimation
#> Newton-Raphson maximisation, 11 iterations
#> Return code 1: gradient close to zero (gradtol)
#> Log-Likelihood: -895.8201
#> 500 observations: 172 selection 1 (FALSE) and 328 selection 2 (TRUE)
#> 10 free parameters (df = 490)
#> Probit selection equation:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  -0.1550     0.1051  -1.474    0.141
#> xs            1.1408     0.1785   6.390 3.86e-10 ***
#> Outcome equation 1:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  0.02708    0.16395   0.165    0.869
#> xo1          0.83959    0.14968   5.609  3.4e-08 ***
#> Outcome equation 2:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   0.1583     0.1885   0.840    0.401
#> xo2           0.8375     0.1707   4.908 1.26e-06 ***
#>    Error terms:
#>        Estimate Std. Error t value Pr(>|t|)
#> sigma1  0.93191    0.09211  10.118   <2e-16 ***
#> sigma2  0.90697    0.04434  20.455   <2e-16 ***
#> rho1    0.88988    0.05353  16.623   <2e-16 ***
#> rho2    0.17695    0.33139   0.534    0.594
```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> --------------------------------------------
```

All the estimates are close to the true values.

Example of functional form misspecification

```
set.seed(5)
eps <- rmvnorm(1000, rep(0, 3), vc)
eps <- eps^2 - 1 # subtract 1 in order to get the mean zero disturbances

# interval [-1, 0] to get an asymmetric distribution over observed choices
xs <- runif(1000, -1, 0)
ys <- xs + eps[,1] > 0
xo1 <- runif(1000)
yo1 <- xo1 + eps[,2]
xo2 <- runif(1000)
yo2 <- xo2 + eps[,3]
summary(selection(ys~xs, list(yo1 ~ xo1, yo2 ~ xo2), iterlim=20))
#> --------------------------------------------
#> Tobit 5 model (switching regression model)
#> Maximum Likelihood estimation
#> Newton-Raphson maximisation, 4 iterations
#> Return code 3: Last step could not find a value above the current.
#> Boundary of parameter space?
#> Consider switching to a more robust optimisation method temporarily.
#> Log-Likelihood: -1665.936
#> 1000 observations: 760 selection 1 (FALSE) and 240 selection 2 (TRUE)
#> 10 free parameters (df = 990)
#> Probit selection equation:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.53698    0.05808  -9.245  < 2e-16 ***
#> xs           0.31268    0.09395   3.328 0.000906 ***
#> Outcome equation 1:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.70679    0.03573  -19.78   <2e-16 ***
#> xo1          0.91603    0.05626   16.28   <2e-16 ***
#> Outcome equation 2:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   0.1446        NaN     NaN      NaN
#> xo2           1.1196     0.5014   2.233   0.0258 *
#>    Error terms:
#>        Estimate Std. Error t value Pr(>|t|)
#> sigma1  0.67770    0.01760   38.50   <2e-16 ***
#> sigma2  2.31432    0.07615   30.39   <2e-16 ***
```

```
#> rho1   -0.97137        NaN     NaN      NaN
#> rho2    0.17039        NaN     NaN      NaN
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> --------------------------------------------
```

Although we still have an exclusion restriction (xo1 and xo2 are independent), we now have problems with the intercepts (i.e., they are statistically significantly different from the true values zero), and convergence problems.

If we don't have the exclusion restriction, we will have a larger variance of xs

```
set.seed(6)
xs <- runif(1000, -1, 1)
ys <- xs + eps[,1] > 0
yo1 <- xs + eps[,2]
yo2 <- xs + eps[,3]
summary(tmp <- selection(ys~xs, list(yo1 ~ xs, yo2 ~ xs), iterlim=20))
#> --------------------------------------------
#> Tobit 5 model (switching regression model)
#> Maximum Likelihood estimation
#> Newton-Raphson maximisation, 16 iterations
#> Return code 8: successive function values within relative tolerance limit (reltol)
#> Log-Likelihood: -1936.431
#> 1000 observations: 626 selection 1 (FALSE) and 374 selection 2 (TRUE)
#> 10 free parameters (df = 990)
#> Probit selection equation:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  -0.3528     0.0424  -8.321 2.86e-16 ***
#> xs            0.8354     0.0756  11.050  < 2e-16 ***
#> Outcome equation 1:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -0.55448    0.06339  -8.748   <2e-16 ***
#> xs           0.81764    0.06048  13.519   <2e-16 ***
#> Outcome equation 2:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   0.6457     0.4994   1.293    0.196
#> xs            0.3520     0.3197   1.101    0.271
#>    Error terms:
#>        Estimate Std. Error t value Pr(>|t|)
#> sigma1  0.59187    0.01853  31.935   <2e-16 ***
#> sigma2  1.97257    0.07228  27.289   <2e-16 ***
#> rho1    0.15568    0.15914   0.978    0.328
#> rho2   -0.01541    0.23370  -0.066    0.947
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#> ------------------------------------------
```

Usually it will not converge. Even if it does, the results may be seriously biased.

Note

The log-likelihood function of the models might not be globally concave. Hence, it might not converge, or converge to a local maximum. To combat this, we can use

- Different starting value
- Different maximization methods.
- refer to [Non-linear Least Squares] for suggestions.

36.3.3.0.1 Pattern-Mixture Models

- compared to the Heckman's model where it assumes the value of the missing data is predetermined, pattern-mixture models assume missingness affect the distribution of variable of interest (e.g., Y)
- To read more, you can check NCSU, stefvanbuuren.

Chapter 37

Other Biases

In econometrics, the main objective is often to uncover causal relationships. However, coefficient estimates can be affected by various biases. Here's a list of common biases that can affect coefficient estimates:

What we've covered so far (see Linear Regression and Endogeneity):

1. **Omitted Variable Bias (OVB)**:
   - Arises when a variable that affects the dependent variable and is correlated with an independent variable is left out of the regression.
2. **Endogeneity Bias**:
   - Occurs when an error term is correlated with an independent variable. This can be due to:
     - Simultaneity: When the dependent variable simultaneously affects an independent variable.
     - Omitted variables.
     - Measurement error in the independent variable.
3. **Measurement Error**:
   - Bias introduced when variables in a model are measured with error. If the error is in an independent variable and is classical (mean zero and uncorrelated with the true value), it typically biases the coefficient towards zero.
4. **Sample Selection Bias**:
   - Arises when the sample is not randomly selected and the selection is related to the dependent variable. A classic example is the Heckman

correction for labor market studies where participants self-select into the workforce.

5. **Simultaneity Bias (or Reverse Causality)**:
   - Happens when the dependent variable causes changes in the independent variable, leading to a two-way causation.
6. **Multicollinearity**:
   - Not a bias in the strictest sense, but in the presence of high multicollinearity (when independent variables are highly correlated), coefficient estimates can become unstable and standard errors large. This makes it hard to determine the individual effect of predictors on the dependent variable.
7. **Specification Errors**:
   - Arise when the functional form of the model is incorrectly specified, e.g., omitting interaction terms or polynomial terms when they are needed.
8. **Autocorrelation (or Serial Correlation)**:
   - Occurs in time-series data when the error terms are correlated over time. This doesn't cause bias in the coefficient estimates of OLS, but it can make standard errors biased, leading to incorrect inference.
9. **Heteroskedasticity**:
   - Occurs when the variance of the error term is not constant across observations. Like autocorrelation, heteroskedasticity doesn't bias the OLS estimates but can bias standard errors.

In this section, we will mention other biases that you may encounter when conducting your research

10.
   - Introduced when data are aggregated, and analysis is conducted at this aggregate level rather than the individual level.
11. [**Survivorship Bias**] **(very much related to Sample Selection)**:
   - Arises when the sample only includes "survivors" or those who "passed" a certain threshold. Common in finance where only funds or firms that "survive" are analyzed.
12.
   - Not a bias in econometric estimation per se, but relevant in the context of empirical studies. It refers to the tendency for journals to publish only significant or positive results, leading to an overrepresentation of such results in the literature.

37.1 Aggregation Bias

Aggregation bias, also known as ecological fallacy, refers to the error introduced when data are aggregated and an analysis is conducted at this aggregate level, rather than at the individual level. This can be especially problematic in econometrics, where analysts are often concerned with understanding individual behavior.

When the relationship between variables is different at the aggregate level than at the individual level, aggregation bias can result. The bias arises when inferences about individual behaviors are made based on aggregate data.

Example: Suppose we have data on individuals' incomes and their personal consumption. At the individual level, it's possible that as income rises, consumption also rises. However, when we aggregate the data to, say, a neighborhood level, neighborhoods with diverse income levels might all have similar average consumption due to other unobserved factors.

Step 1: Create individual level data

```
set.seed(123)

# Generate data for 1000 individuals
n <- 1000

income <- rnorm(n, mean = 50, sd = 10)
consumption <- 0.5 * income + rnorm(n, mean = 0, sd = 5)

# Individual level regression
individual_lm <- lm(consumption ~ income)
summary(individual_lm)
#>
#> Call:
#> lm(formula = consumption ~ income)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -15.1394  -3.4572   0.0213   3.5436  16.4557
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -1.99596    0.82085  -2.432   0.0152 *
#> income       0.54402    0.01605  33.888   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 5.032 on 998 degrees of freedom
#> Multiple R-squared:  0.535,  Adjusted R-squared:  0.5346
```

```
#> F-statistic:  1148 on 1 and 998 DF,  p-value: < 2.2e-16
```

This would show a significant positive relationship between income and consumption.

Step 2: Aggregate data to 'neighborhood' level

```
# Assume 100 neighborhoods with 10 individuals each
n_neighborhoods <- 100

df <- data.frame(income, consumption)
df$neighborhood <- rep(1:n_neighborhoods, each = n / n_neighborhoods)

aggregate_data <- aggregate(. ~ neighborhood, data = df, FUN = mean)

# Aggregate level regression
aggregate_lm <- lm(consumption ~ income, data = aggregate_data)
summary(aggregate_lm)
#>
#> Call:
#> lm(formula = consumption ~ income, data = aggregate_data)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -4.4517 -0.9322 -0.0826  1.0556  3.5728
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -4.94338    2.60699  -1.896   0.0609 .
#> income       0.60278    0.05188  11.618   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.54 on 98 degrees of freedom
#> Multiple R-squared:  0.5794, Adjusted R-squared:  0.5751
#> F-statistic:   135 on 1 and 98 DF,  p-value: < 2.2e-16
```

If aggregation bias is present, the coefficient for income in the aggregate regression might be different from the coefficient in the individual regression, even if the individual relationship is significant and strong.

```
library(ggplot2)

# Individual scatterplot
p1 <- ggplot(df, aes(x = income, y = consumption)) +
    geom_point(aes(color = neighborhood), alpha = 0.6) +
    geom_smooth(method = "lm",
```

```
                 se = FALSE,
                 color = "black") +
    labs(title = "Individual Level Data") +
    causalverse::ama_theme()

# Aggregate scatterplot
p2 <- ggplot(aggregate_data, aes(x = income, y = consumption)) +
    geom_point(color = "red") +
    geom_smooth(method = "lm",
                se = FALSE,
                color = "black") +
    labs(title = "Aggregate Level Data") +
    causalverse::ama_theme()

# print(p1)
# print(p2)

gridExtra::grid.arrange(grobs = list(p1, p2), ncol = 2)
```

From these plots, you can see the relationship at the individual level, with each neighborhood being colored differently in the first plot. The second plot shows the aggregate data, where each point now represents a whole neighborhood.

Direction of Bias: The direction of the aggregation bias isn't predetermined. It depends on the underlying relationship and the data distribution. In some cases, aggregation might attenuate (reduce) a relationship, while in other cases, it might exaggerate it.

Relation to Other Biases: Aggregation bias is closely related to several other biases in econometrics:

1. **Specification bias**: If you don't properly account for the hierarchical structure of your data (like individuals nested within neighborhoods), your model might be mis-specified, leading to biased estimates.
2. **Measurement Error**: Aggregation can introduce or amplify measurement errors. For instance, if you aggregate noisy measures, the aggregate might not accurately represent any underlying signal.
3. **Omitted Variable Bias (see Endogeneity)**: When you aggregate data, you lose information. If the loss of this information results in omitting important predictors that are correlated with both the independent and dependent variables, it can introduce omitted variable bias.

37.1.1 Simpson's Paradox

Simpson's Paradox, also known as the Yule-Simpson effect, is a phenomenon in probability and statistics where a trend that appears in different groups of data disappears or reverses when the groups are combined. It's a striking example of how aggregated data can sometimes provide a misleading representation of the actual situation.

Illustration of Simpson's Paradox:

Consider a hypothetical scenario involving two hospitals: Hospital A and Hospital B. We want to analyze the success rates of treatments at both hospitals. When we break the data down by the severity of the cases (i.e., minor cases vs. major cases):

1. **Hospital A**:
   - Minor cases: 95% success rate
   - Major cases: 80% success rate
2. **Hospital B**:
   - Minor cases: 90% success rate
   - Major cases: 85% success rate

From this breakdown, Hospital A appears to be better in treating both minor and major cases since it has a higher success rate in both categories.

However, let's consider the overall success rates without considering case severity:

- **Hospital A**: 83% overall success rate
- **Hospital B**: 86% overall success rate

Suddenly, Hospital B seems better overall. This surprising reversal happens because the two hospitals might handle very different proportions of minor and major cases. For example, if Hospital A treats many more major cases (which have lower success rates) than Hospital B, it can drag down its overall success rate.

Causes:

Simpson's Paradox can arise due to:

1. A lurking or confounding variable that wasn't initially considered (in our example, the severity of the medical cases).
2. Different group sizes, where one group might be much larger than the other, influencing the aggregate results.

Implications:

Simpson's Paradox highlights the dangers of interpreting aggregated data without considering potential underlying sub-group structures. It underscores the importance of disaggregating data and being aware of the context in which it's analyzed.

Relation to Aggregation Bias

In the most extreme case, aggregation bias can reverse the coefficient sign of the relationship of interest (i.e., Simpson's Paradox).

Example: Suppose we are studying the effect of a new study technique on student grades. We have two groups of students: those who used the new technique (`treatment = 1`) and those who did not (`treatment = 0`). We want to see if using the new study technique is related to higher grades.

Let's assume grades are influenced by the starting ability of the students. Perhaps in our sample, many high-ability students didn't use the new technique (because they felt they didn't need it), while many low-ability students did.

Here's a setup:

1. High-ability students tend to have high grades regardless of the technique.
2. The new technique has a positive effect on grades, but this is masked by the fact that many low-ability students use it.

```r
set.seed(123)

# Generate data for 1000 students
n <- 1000

# 500 students are of high ability, 500 of low ability
ability <- c(rep("high", 500), rep("low", 500))
```

```r
# High ability students are less likely to use the technique
treatment <-
  ifelse(ability == "high", rbinom(500, 1, 0.2), rbinom(500, 1, 0.8))

# Grades are influenced by ability and treatment (new technique),
# but the treatment has opposite effects based on ability.
grades <-
  ifelse(
    ability == "high",
    rnorm(500, mean = 85, sd = 5) + treatment * -3,
    rnorm(500, mean = 60, sd = 5) + treatment * 5
  )

df <- data.frame(ability, treatment, grades)

# Regression without considering ability
overall_lm <- lm(grades ~ factor(treatment), data = df)
summary(overall_lm)
#>
#> Call:
#> lm(formula = grades ~ factor(treatment), data = df)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -33.490  -4.729   0.986   6.368  25.607
#>
#> Coefficients:
#>                     Estimate Std. Error t value Pr(>|t|)
#> (Intercept)          80.0133     0.4373   183.0   <2e-16 ***
#> factor(treatment)1 -11.7461     0.6248   -18.8   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 9.877 on 998 degrees of freedom
#> Multiple R-squared:  0.2615, Adjusted R-squared:  0.2608
#> F-statistic: 353.5 on 1 and 998 DF,  p-value: < 2.2e-16

# Regression within ability groups
high_ability_lm <-
  lm(grades ~ factor(treatment), data = df[df$ability == "high",])
low_ability_lm <-
  lm(grades ~ factor(treatment), data = df[df$ability == "low",])
summary(high_ability_lm)
#>
#> Call:
```

```
#> lm(formula = grades ~ factor(treatment), data = df[df$ability ==
#>     "high", ])
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -14.2156  -3.4813   0.1186   3.4952  13.2919
#>
#> Coefficients:
#>                    Estimate Std. Error t value Pr(>|t|)
#> (Intercept)         85.1667     0.2504 340.088  < 2e-16 ***
#> factor(treatment)1  -3.9489     0.5776  -6.837 2.37e-11 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 5.046 on 498 degrees of freedom
#> Multiple R-squared:  0.08581,    Adjusted R-squared:  0.08398
#> F-statistic: 46.75 on 1 and 498 DF,  p-value: 2.373e-11
summary(low_ability_lm)
#>
#> Call:
#> lm(formula = grades ~ factor(treatment), data = df[df$ability ==
#>     "low", ])
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -13.3717  -3.5413   0.1097   3.3531  17.0568
#>
#> Coefficients:
#>                    Estimate Std. Error t value Pr(>|t|)
#> (Intercept)         59.8950     0.4871 122.956   <2e-16 ***
#> factor(treatment)1   5.2979     0.5474   9.679   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 4.968 on 498 degrees of freedom
#> Multiple R-squared:  0.1583, Adjusted R-squared:  0.1566
#> F-statistic: 93.68 on 1 and 498 DF,  p-value: < 2.2e-16
```

From this simulation:

- The **`overall_lm`** might show that the new study technique is associated with lower grades (negative coefficient), because many of the high-ability students (who naturally have high grades) did not use it.
- The **`high_ability_lm`** will likely show that high-ability students who used the technique had slightly lower grades than high-ability students who

didn't.

- The **low_ability_lm** will likely show that low-ability students who used the technique had much higher grades than low-ability students who didn't.

This is a classic example of Simpson's Paradox: within each ability group, the technique appears beneficial, but when data is aggregated, the effect seems negative because of the distribution of the technique across ability groups.

```
library(ggplot2)

# Scatterplot for overall data
p1 <-
  ggplot(df, aes(
    x = factor(treatment),
    y = grades,
    color = ability
  )) +
  geom_jitter(width = 0.2, height = 0) +
  geom_boxplot(alpha = 0.6, outlier.shape = NA) +
  labs(title = "Overall Effect of Study Technique on Grades",
       x = "Treatment (0 = No Technique, 1 = New Technique)",
       y = "Grades")

# Scatterplot for high-ability students
p2 <-
  ggplot(df[df$ability == "high", ], aes(
    x = factor(treatment),
    y = grades,
    color = ability
  )) +
  geom_jitter(width = 0.2, height = 0) +
  geom_boxplot(alpha = 0.6, outlier.shape = NA) +
  labs(title = "Effect of Study Technique on Grades (High Ability)",
       x = "Treatment (0 = No Technique, 1 = New Technique)",
       y = "Grades")

# Scatterplot for low-ability students
p3 <-
  ggplot(df[df$ability == "low", ], aes(
    x = factor(treatment),
    y = grades,
    color = ability
  )) +
  geom_jitter(width = 0.2, height = 0) +
  geom_boxplot(alpha = 0.6, outlier.shape = NA) +
```

```
  labs(title = "Effect of Study Technique on Grades (Low Ability)",
       x = "Treatment (0 = No Technique, 1 = New Technique)",
       y = "Grades")

# print(p1)
# print(p2)
# print(p3)
gridExtra::grid.arrange(grobs = list(p1, p2, p3), ncol = 1)
```

37.2 Contamination Bias

Goldsmith-Pinkham et al. (2022) show regressions with multiple treatments and flexible controls often fail to estimate convex averages of heterogeneous treatment effects, resulting in contamination by non-convex averages of other treatments' effects.

- 3 estimation methods to avoid this bias and find significant contamination bias in observational studies, with experimental studies showing less due to lower variability in propensity scores.

37.3 Survivorship Bias

Survivorship bias refers to the logical error of concentrating on the entities that have made it past some selection process and overlooking those that didn't, typically because of a lack of visibility. This can skew results and lead to overly optimistic conclusions.

Example: If you were to analyze the success of companies based only on the ones that are still in business today, you'd miss out on the insights from all those that failed. This would give you a distorted view of what makes a successful company, as you wouldn't account for all those that had those same attributes but didn't succeed.

Relation to Other Biases:

1. **Sample Selection Bias**: Survivorship bias is a specific form of sample selection bias. While survivorship bias focuses on entities that "survive", sample selection bias broadly deals with any non-random sample.

2. **Confirmation Bias**: Survivorship bias can reinforce confirmation bias. By only looking at the "winners", we might confirm our existing beliefs about what leads to success, ignoring evidence to the contrary from those that didn't survive.

```r
set.seed(42)

# Generating data for 100 companies
n <- 100

# Randomly generate earnings; assume true average earnings is 50
earnings <- rnorm(n, mean = 50, sd = 10)

# Threshold for bankruptcy
threshold <- 40

# Only companies with earnings above the threshold "survive"
survivor_earnings <- earnings[earnings > threshold]

# Average earnings for all companies vs. survivors
true_avg <- mean(earnings)
survivor_avg <- mean(survivor_earnings)

true_avg
#> [1] 50.32515
survivor_avg
#> [1] 53.3898
```

Using a histogram to visualize the distribution of earnings, highlighting the "survivors".

```r
library(ggplot2)

df <- data.frame(earnings)

p <- ggplot(df, aes(x = earnings)) +
```

```
  geom_histogram(
    binwidth = 2,
    fill = "grey",
    color = "black",
    alpha = 0.7
  ) +
  geom_vline(aes(xintercept = true_avg, color = "True Avg"),
             linetype = "dashed",
             size = 1) +
  geom_vline(
    aes(xintercept = survivor_avg, color = "Survivor Avg"),
    linetype = "dashed",
    size = 1
  ) +
  scale_color_manual(values = c("True Avg" = "blue", "Survivor Avg" = "red"),
                     name = "Average Type") +
  labs(title = "Distribution of Company Earnings",
       x = "Earnings",
       y = "Number of Companies") +
  causalverse::ama_theme()

print(p)
```

In the plot, the "True Avg" might be lower than the "Survivor Avg", indicating that by only looking at the survivors, we overestimate the average earnings.

Remedies:

1. **Awareness**: Recognizing the potential for survivorship bias is the first step.
2. **Inclusive Data Collection**: Wherever possible, try to include data from entities that didn't "survive" in your sample.
3. **Statistical Techniques**: In cases where the missing data is inherent, methods like Heckman's two-step procedure can be used to correct for sample selection bias.
4. **External Data Sources**: Sometimes, complementary datasets can provide insights into the missing "non-survivors".
5. **Sensitivity Analysis**: Test how sensitive your results are to assumptions about the non-survivors.

37.4 Publication Bias

Publication bias occurs when the results of studies influence the likelihood of their being published. Typically, studies with significant, positive, or sensational results are more likely to be published than those with non-significant or negative results. This can skew the perceived effectiveness or results when researchers conduct meta-analyses or literature reviews, leading them to draw inaccurate conclusions.

Example: Imagine pharmaceutical research. If 10 studies are done on a new drug, and only 2 show a positive effect while 8 show no effect, but only the 2 positive studies get published, a later review of the literature might erroneously conclude the drug is effective.

Relation to Other Biases:

1. **Selection Bias**: Publication bias is a form of selection bias, where the selection (publication in this case) isn't random but based on the results of the study.
2. **Confirmation Bias**: Like survivorship bias, publication bias can reinforce confirmation bias. Researchers might only find and cite studies that confirm their beliefs, overlooking the unpublished studies that might contradict them.

Let's simulate an experiment on a new treatment. We'll assume that the treatment has no effect, but due to random variation, some studies will show significant positive or negative effects.

```r
set.seed(42)

# Number of studies
n <- 100
```

```r
# Assuming no real effect (effect size = 0)
true_effect <- 0

# Random variation in results
results <- rnorm(n, mean = true_effect, sd = 1)

# Only "significant" results get published
# (arbitrarily defining significant as abs(effect) > 1.5)
published_results <- results[abs(results) > 1.5]

# Average effect for all studies vs. published studies
true_avg_effect <- mean(results)
published_avg_effect <- mean(published_results)

true_avg_effect
#> [1] 0.03251482
published_avg_effect
#> [1] -0.3819601
```

Using a histogram to visualize the distribution of study results, highlighting the "published" studies.

```r
library(ggplot2)

df <- data.frame(results)

p <- ggplot(df, aes(x = results)) +
  geom_histogram(
    binwidth = 0.2,
    fill = "grey",
    color = "black",
    alpha = 0.7
  ) +
  geom_vline(
    aes(xintercept = true_avg_effect,
        color = "True Avg Effect"),
    linetype = "dashed",
    size = 1
  ) +
  geom_vline(
    aes(xintercept = published_avg_effect,
        color = "Published Avg Effect"),
    linetype = "dashed",
    size = 1
  ) +
  scale_color_manual(
```

```
    values = c(
      "True Avg Effect" = "blue",
      "Published Avg Effect" = "red"
    ),
    name = "Effect Type"
  ) +
  labs(title = "Distribution of Study Results",
       x = "Effect Size",
       y = "Number of Studies") +
  causalverse::ama_theme()

print(p)
```

The plot might show that the "True Avg Effect" is around zero, while the "Published Avg Effect" is likely higher or lower, depending on which studies happen to have significant results in the simulation.

Remedies:

1. **Awareness**: Understand and accept that publication bias exists, especially when conducting literature reviews or meta-analyses.
2. **Study Registries**: Encourage the use of study registries where researchers register their studies before they start. This way, one can see all initiated studies, not just the published ones.
3. **Publish All Results**: Journals and researchers should make an effort to publish negative or null results. Some journals, known as "null result journals", specialize in this.

4. **Funnel Plots and Egger's Test**: In meta-analyses, these are methods to visually and statistically detect publication bias.

5. **Use of Preprints**: Promote the use of preprint servers where researchers can upload studies before they're peer-reviewed, ensuring that results are available regardless of eventual publication status.

6. **p-curve analysis**: addresses publication bias and p-hacking by analyzing the distribution of p-values below 0.05 in research studies. It posits that a right-skewed distribution of these p-values indicates a true effect, whereas a left-skewed distribution suggests p-hacking and no true underlying effect. The method includes a "half-curve" test to counteract extensive p-hacking (Simonsohn et al., 2014a, Simonsohn et al. (2014b), Simonsohn et al. (2015)).

Chapter 38

Controls

This section follows (Cinelli et al., 2022) and code

```
library(dagitty)
library(ggdag)
```

Traditional literature usually considers adding additional control variables is harmless to analysis.

More specifically, this problem is most prevalent in the review process. Reviewers only ask authors to add more variables to "control" for such variable, which can be asked with only limited rationale. Rarely ever you will see a reviewer asks an author to remove some variables to see the behavior of the variable of interest (This is also related to Coefficient stability).

However, adding more controls is only good in limited cases.

38.1 Bad Controls

38.1.1 M-bias

Traditional textbooks (Imbens and Rubin, 2015; Angrist and Pischke, 2009) consider Z as a good control because it's a pre-treatment variable, where it correlates with the treatment and the outcome.

This is most prevalent in Matching Methods, where we are recommended to include all "pre-treatment" variables.

However, it is a bad control because it opens the back-door path $Z \leftarrow U_1 \rightarrow Z \leftarrow U_2 \rightarrow Y$

```
# cleans workspace
rm(list = ls())
```

```r
# DAG

## specify edges
model <- dagitty("dag{x->y; u1->x; u1->z; u2->z; u2->y}")

# set u as latent
latents(model) <- c("u1", "u2")

## coordinates for plotting
coordinates(model) <-  list(x = c(
    x = 1,
    u1 = 1,
    z = 2,
    u2 = 3,
    y = 3
),
y = c(
    x = 1,
    u1 = 2,
    z = 1.5,
    u2 = 2,
    y = 1
))

## ggplot
ggdag(model) + theme_dag()
```

Even though Z can correlate with both X and Y very well, it's not a confounder.

Controlling for Z can bias the $X \to Y$ estimate, because it opens the colliding path $X \leftarrow U_1 \to Z \leftarrow U_2 \leftarrow Y$

```
n <- 1e4
u1 <- rnorm(n)
u2 <- rnorm(n)
z <- u1 + u2 + rnorm(n)
x <- u1 + rnorm(n)
causal_coef <- 2
y <- causal_coef * x - 4*u2 + rnorm(n)


jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

Table 38.1

	Model 1	Model 2
(Intercept)	-0.00	-0.01
	(0.04)	(0.03)
x	1.95 ***	2.78 ***
	(0.03)	(0.03)
z		-1.65 ***
		(0.02)
N	10000	10000
R2	0.30	0.57

*** p < 0.001; ** p < 0.01; * p < 0.05.

Another worse variation is

```
# cleans workspace
rm(list = ls())

# DAG

## specify edges
model <- dagitty("dag{x->y; u1->x; u1->z; u2->z; u2->y; z->y}")
```

```r
# set u as latent
latents(model) <- c("u1", "u2")

## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, u1=1, z=2, u2=3, y=3),
  y = c(x=1, u1=2, z=1.5, u2=2, y=1))

## ggplot
ggdag(model) + theme_dag()
```

You can't do much in this case.

- If you don't control for Z, then you have an open back-door path $X \leftarrow U_1 \rightarrow Z \rightarrow Y$, and the unadjusted estimate is biased

- If you control for Z, then you open backdoor path $X \leftarrow U_1 \rightarrow Z \leftarrow U_2 \rightarrow Y$, and the adjusted estimate is also biased

Hence, we cannot identify the causal effect in this case.

We can do sensitivity analyses to examine (Cinelli et al., 2019; Cinelli and Hazlett, 2020)

1. the plausible bounds on the strength of the direct effect of $Z \rightarrow Y$
2. the strength of the effects of the latent variables

38.1.2 Bias Amplification

```
# cleans workspace
rm(list = ls())

# DAG

## specify edges
model <- dagitty("dag{x->y; u->x; u->y; z->x}")

# set u as latent
latents(model) <- c("u")

## coordinates for plotting
coordinates(model) <-  list(
  x = c(z=1, x=2, u=3, y=4),
  y = c(z=1, x=1, u=2, y=1))

## ggplot
ggdag(model) + theme_dag()
```

Controlling for Z amplifies the omitted variable bias

```
n <- 1e4
z <- rnorm(n)
u <- rnorm(n)
```

```
x <- 2*z + u + rnorm(n)
y <- x + 2*u + rnorm(n)

jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

Table 38.2

	Model 1	Model 2
(Intercept)	0.03	0.01
	(0.02)	(0.02)
x	1.33 ***	1.99 ***
	(0.01)	(0.01)
z		-1.97 ***
		(0.03)
N	10000	10000
R2	0.71	0.80

*** p < 0.001; ** p < 0.01; * p < 0.05.

38.1.3 Overcontrol bias

Sometimes, this is similar to controlling for variables that are proxy of the dependent variable.

```
# cleans workspace
rm(list = ls())

# DAG

## specify edges
model <- dagitty("dag{x->z; z->y}")


## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, z=2, y=3),
  y = c(x=1, z=1, y=1))
```

```
## ggplot
ggdag(model) + theme_dag()
```

If X is a proxy for Z (i.e., a mediator between Z and Y), controlling for Z is bad

```
n <- 1e4
x <- rnorm(n)
z <- x + rnorm(n)
y <- z + rnorm(n)

jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

Now you see that Z is significant, which is technically true, but we are interested in the causal coefficient of X on Y.

Another setting for overcontrol bias is

```
# cleans workspace
rm(list = ls())

# DAG

## specify edges
model <- dagitty("dag{x->m; m->z; m->y}")


## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, m=2, z=2, y=3),
```

Table 38.3

	Model 1	Model 2
(Intercept)	0.01	0.02
	(0.01)	(0.01)
x	0.99 ***	0.01
	(0.01)	(0.01)
z		1.00 ***
		(0.01)
N	10000	10000
R2	0.33	0.66

*** p < 0.001; ** p < 0.01; * p < 0.05.

```
  y = c(x=2, m=2, z=1, y=2))

## ggplot
ggdag(model) + theme_dag()
```

```r
n <- 1e4
x <- rnorm(n)
m <- x + rnorm(n)
z <- m + rnorm(n)
y <- m + rnorm(n)


jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

Table 38.4

	Model 1	Model 2
(Intercept)	0.00	-0.00
	(0.01)	(0.01)
x	0.98 ***	0.48 ***
	(0.01)	(0.01)
z		0.50 ***
		(0.01)
N	10000	10000
R2	0.34	0.51

*** p < 0.001; ** p < 0.01; * p < 0.05.

Another setting for this bias is

```r
# cleans workspace
rm(list = ls())

# DAG

## specify edges
model <- dagitty("dag{x->z; z->y; u->z; u->y}")

# set u as latent
latents(model) <- "u"

## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, z=2, u=3, y=4),
```

```
  y = c(x=1, z=1, u=2, y=1))

## ggplot
ggdag(model) + theme_dag()
```

```
set.seed(1)
n <- 1e4
x <- rnorm(n)
u <- rnorm(n)
z <- x + u + rnorm(n)
y <- z + u + rnorm(n)

jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

The total effect of X on Y is not biased (i.e., $1.01 \approx 1.48 - 0.47$).

Controlling for Z will fail to identify the direct effect of X on Y and opens the biasing path $X \to Z \leftarrow U \to Y$

38.1.4 Selection Bias

Also known as "collider stratification bias"

```
rm(list = ls())

# DAG

## specify edges
```

Table 38.5

	Model 1	Model 2
(Intercept)	-0.01	-0.01
	(0.02)	(0.01)
x	1.01 ***	-0.47 ***
	(0.02)	(0.01)
z		1.48 ***
		(0.01)
N	10000	10000
R2	0.15	0.78

*** p < 0.001; ** p < 0.01; * p < 0.05.

```
model <- dagitty("dag{x->y; x->z; u->z;u->y}")

# set u as latent
latents(model) <- "u"

## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, z=2, u=2, y=3),
  y = c(x=3, z=2, u=4, y=3))

## ggplot
ggdag(model) + theme_dag()
```

Adjusting Z opens the colliding path $X \to Z \leftarrow U \to Y$

```
n <- 1e4
x <- rnorm(n)
u <- rnorm(n)
z <- x + u +  rnorm(n)
y <- x + 2*u + rnorm(n)

jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

Another setting is

```
rm(list = ls())

# DAG

## specify edges
model <- dagitty("dag{x->y; x->z; y->z}")

## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, z=2, y=3),
  y = c(x=2, z=1, y=2))

## ggplot
ggdag(model) + theme_dag()
```

Table 38.6

	Model 1	Model 2
(Intercept)	-0.01	0.01
	(0.02)	(0.02)
x	0.97 ***	-0.03
	(0.02)	(0.02)
z		1.00 ***
		(0.01)
N	10000	10000
R2	0.16	0.49

*** p < 0.001; ** p < 0.01; * p < 0.05.

Controlling Z opens the colliding path $X \to Z \leftarrow Y$

```
n <- 1e4
x <- rnorm(n)
y <- x + rnorm(n)
z <- x + y + rnorm(n)
```

```
jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

Table 38.7

	Model 1	Model 2
(Intercept)	0.00	0.00
	(0.01)	(0.01)
x	1.03 ***	-0.00
	(0.01)	(0.01)
z		0.51 ***
		(0.00)
N	10000	10000
R2	0.51	0.76

*** p < 0.001; ** p < 0.01; * p < 0.05.

```
rm(list = ls())

# DAG

## specify edges
model <- dagitty("dag{x->y; y->z}")

## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, z=2, y=3),
  y = c(x=2, z=1, y=2))

## ggplot
ggdag(model) + theme_dag()
```

Controlling Z opens a virtual collider (a descendant of a collider).

However, if X truly has no causal effect on Y. Then, controlling for Z is valid for testing whether the effect of X on Y is 0 because X is d-separated from Y regardless of adjusting for Z

```
n <- 1e4
x <- rnorm(n)
y <- x + rnorm(n)
z <- y + rnorm(n)

jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

38.2 Good Controls

38.2.1 Omitted Variable Bias Correction

This is when Z can block all back-door paths.

```
rm(list = ls())

model <- dagitty("dag{x->y; z->x; z->y}")

## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, y=3, z=2),
  y = c(x=1, y=1, z=2))
```

Table 38.8

	Model 1	Model 2
(Intercept)	-0.00	-0.00
	(0.01)	(0.01)
x	1.00 ***	0.50 ***
	(0.01)	(0.01)
z		0.50 ***
		(0.00)
N	10000	10000
R2	0.50	0.75

*** p < 0.001; ** p < 0.01; * p < 0.05.

```
## ggplot
ggdag(model) + theme_dag()
```

Unadjusted estimate is biased

adjusting for Z blocks the backdoor path

```r
n <- 1e4
z <- rnorm(n)
causal_coef = 2
beta2 = 3
x <- z + rnorm(n)
y <- causal_coef * x + beta2 * z + rnorm(n)

jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

Table 38.9

	Model 1	Model 2
(Intercept)	0.01	-0.00
	(0.02)	(0.01)
x	3.51 ***	2.00 ***
	(0.02)	(0.01)
z		3.02 ***
		(0.01)
N	10000	10000
R2	0.82	0.97

*** p < 0.001; ** p < 0.01; * p < 0.05.

```r
# cleans workspace
rm(list = ls())

# Draw DAG

# specify edges
model <- dagitty("dag{x->y; u->z; z->x; u->y}")

# set u as latent
latents(model) <- "u"

## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, z=2, u=3, y = 4),
  y = c(x=1, y=1, z=2, u = 3))
```

```r
## ggplot
ggdag(model) + theme_dag()
```

Unadjusted estimate is biased

adjusting for Z blocks the backdoor door path due to U

```r
n <- 1e4
u <- rnorm(n)
z <- u + rnorm(n)
causal_coef = 2
x <- z + rnorm(n)
y <- causal_coef * x + u + rnorm(n)

jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

Even though Z is significant, we cannot give it a causal interpretation.

```r
# cleans workspace
rm(list = ls())

# Draw DAG

# specify edges
model <- dagitty("dag{x->y; u->z; u->x; z->y}")

# set u as latent
latents(model) <- "u"
```

Table 38.10

	Model 1	Model 2
(Intercept)	0.03 *	0.03 *
	(0.01)	(0.01)
x	2.34 ***	2.01 ***
	(0.01)	(0.01)
z		0.49 ***
		(0.01)
N	10000	10000
R2	0.91	0.92

*** p < 0.001; ** p < 0.01; * p < 0.05.

```
## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, z=3, u=2, y = 4),
  y = c(x=1, y=1, z=2, u = 3))

## ggplot
ggdag(model) + theme_dag()
```

```
n     <- 1e4
u     <- rnorm(n)
z     <- u + rnorm(n)
x     <- u + rnorm(n)
causal_coef <- 2
y     <- causal_coef * x + z + rnorm(n)

jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

Even though Z is significant, we cannot give it a causal interpretation.

Summary

```
# cleans workspace
rm(list = ls())

# Model 1

model1 <- dagitty("dag{x->y; z->x; z->y}")

## coordinates for plotting
coordinates(model1) <-  list(
  x = c(x=1, y=3, z=2),
  y = c(x=1, y=1, z=2))



# Model 2
```

Table 38.11

	Model 1	Model 2
(Intercept)	-0.03	-0.01
	(0.02)	(0.01)
x	2.51 ***	2.01 ***
	(0.01)	(0.01)
z		1.01 ***
		(0.01)
N	10000	10000
R2	0.84	0.93

*** p < 0.001; ** p < 0.01; * p < 0.05.

```
# specify edges
model2 <- dagitty("dag{x->y; u->z; z->x; u->y}")

# set u as latent
latents(model2) <- "u"

## coordinates for plotting
coordinates(model2) <-  list(
  x = c(x=1, z=2, u=3, y = 4),
  y = c(x=1, y=1, z=2, u = 3))



# Model 3

# specify edges
model3 <- dagitty("dag{x->y; u->z; u->x; z->y}")

# set u as latent
latents(model3) <- "u"

## coordinates for plotting
coordinates(model3) <-  list(
```

```
  x = c(x=1, z=3, u=2, y = 4),
  y = c(x=1, y=1, z=2, u = 3))

par(mfrow=c(1,3))

## ggplot
ggdag(model1) + theme_dag()
```

```
## ggplot
ggdag(model2) + theme_dag()
```

```
## ggplot
ggdag(model3) + theme_dag()
```

38.2.2 Omitted Variable Bias in Mediation Correction

Common causes of X and any mediator (between X and Y) confound the effect of X on Y

```r
# cleans workspace
rm(list = ls())

# DAG

## specify edges
model <- dagitty("dag{x->y; z->x; x->m; z->m; m->y}")

## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, z=2, m=3, y=4),
  y = c(x=1, z=2, m=1, y=1))

## ggplot
ggdag(model) + theme_dag()
```

Z is a confounder of both the mediator M and X

```r
n     <- 1e4
z     <- rnorm(n)
x     <- z + rnorm(n)
causal_coef <- 2
m     <- causal_coef * x + z + rnorm(n)
y     <- m + rnorm(n)

jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

Table 38.12

	Model 1	Model 2
(Intercept)	-0.02	-0.01
	(0.02)	(0.01)
x	2.49 ***	1.97 ***
	(0.01)	(0.01)
z		1.02 ***
		(0.02)
N	10000	10000
R2	0.83	0.86

*** p < 0.001; ** p < 0.01; * p < 0.05.

```r
# cleans workspace
rm(list = ls())

# DAG

## specify edges
model <- dagitty("dag{x->y; u->z; z->x; x->m; u->m; m->y}")

# set u as latent
latents(model) <- "u"

## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, z=2, u=3, m=4, y=5),
  y = c(x=1, z=2, u=3, m=1, y=1))

## ggplot
ggdag(model) + theme_dag()
```

```
n     <- 1e4
u     <- rnorm(n)
z     <- u + rnorm(n)
x     <- z + rnorm(n)
causal_coef <- 2
m     <- causal_coef * x + u + rnorm(n)
y     <- m + rnorm(n)

jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

```
# cleans workspace
rm(list = ls())

# DAG

## specify edges
model <- dagitty("dag{x->y; u->z; z->m; x->m; u->x; m->y}")

# set u as latent
latents(model) <- "u"

## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, z=3, u=2, m=4, y=5),
  y = c(x=1, z=2, u=3, m=1, y=1))

## ggplot
```

Table 38.13

	Model 1	Model 2
(Intercept)	-0.01	-0.01
	(0.02)	(0.02)
x	2.31 ***	2.00 ***
	(0.01)	(0.02)
z		0.49 ***
		(0.02)
N	10000	10000
R2	0.86	0.86

*** p < 0.001; ** p < 0.01; * p < 0.05.

```
ggdag(model) + theme_dag()
```

```
n     <- 1e4
u     <- rnorm(n)
z     <- u + rnorm(n)
```

```
x      <- u + rnorm(n)
causal_coef <- 2
m      <- causal_coef * x + z + rnorm(n)
y      <- m + rnorm(n)

jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

Table 38.14

	Model 1	Model 2
(Intercept)	0.01	-0.00
	(0.02)	(0.01)
x	2.50 ***	1.99 ***
	(0.01)	(0.01)
z		1.02 ***
		(0.01)
N	10000	10000
R2	0.78	0.87

*** p < 0.001; ** p < 0.01; * p < 0.05.

Summary

```
# model 4
model4 <- dagitty("dag{x->y; z->x; x->m; z->m; m->y}")

## coordinates for plotting
coordinates(model4) <-  list(
  x = c(x=1, z=2, m=3, y=4),
  y = c(x=1, z=2, m=1, y=1))


# model 5
model5 <- dagitty("dag{x->y; u->z; z->x; x->m; u->m; m->y}")

# set u as latent
latents(model5) <- "u"

## coordinates for plotting
```

```
coordinates(model5) <-  list(
  x = c(x=1, z=2, u=3, m=4, y=5),
  y = c(x=1, z=2, u=3, m=1, y=1))


# model 6

model6 <- dagitty("dag{x->y; u->z; z->m; x->m; u->x; m->y}")

# set u as latent
latents(model6) <- "u"

## coordinates for plotting
coordinates(model6) <-  list(
  x = c(x=1, z=3, u=2, m=4, y=5),
  y = c(x=1, z=2, u=3, m=1, y=1))

par(mfrow=c(1,3))

## ggplot
ggdag(model4) + theme_dag()
```

```
## ggplot
ggdag(model5) + theme_dag()
```

```
## ggplot
ggdag(model6) + theme_dag()
```

38.3 Neutral Controls

38.3.1 Good Predictive Controls

Good for precision

```r
# cleans workspace
rm(list = ls())

# DAG

## specify edges
model <- dagitty("dag{x->y; z->y}")


## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, z=2, y=2),
  y = c(x=1, z=2, y=1))

## ggplot
ggdag(model) + theme_dag()
```

Controlling for Z does not help or hurt identification, but it can increase precision (i.e., reducing SE)

```r
n <- 1e4
z <- rnorm(n)
x <- rnorm(n)
y <- x + 2 * z + rnorm(n)

jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

Similar coefficients, but smaller SE when controlling for Z

Table 38.15

	Model 1	Model 2
(Intercept)	0.01	0.01
	(0.02)	(0.01)
x	1.00 ***	1.01 ***
	(0.02)	(0.01)
z		2.00 ***
		(0.01)
N	10000	10000
R2	0.17	0.83

*** p < 0.001; ** p < 0.01; * p < 0.05.

Another variation is

```
# cleans workspace
rm(list = ls())

# DAG

## specify edges
model <- dagitty("dag{x->y; x->m; z->m; m->y}")

## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, z=2, m=2, y=3),
  y = c(x=1, z=2, m=1, y=1))

## ggplot
ggdag(model) + theme_dag()
```

```
n <- 1e4
z <- rnorm(n)
x <- rnorm(n)
m <- 2 * z + rnorm(n)
y <- x + 2 * m + rnorm(n)

jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

Controlling for Z can reduce SE

38.3.2 Good Selection Bias

```
# cleans workspace
rm(list = ls())

# DAG

## specify edges
model <- dagitty("dag{x->y; x->z; z->w; u->w;u->y}")

# set u as latent
latents(model) <- "u"

## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, z=2, w=3, u=3, y=5),
  y = c(x=3, z=2, w=1, u=4, y=3))
```

Table 38.16

	Model 1	Model 2
(Intercept)	-0.00	-0.00
	(0.05)	(0.02)
x	0.97 ***	0.99 ***
	(0.05)	(0.02)
z		4.02 ***
		(0.02)
N	10000	10000
R2	0.04	0.77

*** p < 0.001; ** p < 0.01; * p < 0.05.

```
## ggplot
ggdag(model) + theme_dag()
```

1. Unadjusted estimate is unbiased
2. Controlling for Z can increase SE

3. Controlling for Z while having on W can help identify X

```
n <- 1e4
x <- rnorm(n)
u <- rnorm(n)
z <- x + rnorm(n)
w <- z + u + rnorm(n)
y <- x - 2*u + rnorm(n)

jtools::export_summs(lm(y ~ x), lm(y ~ x + w), lm(y ~ x + z + w))
```

Table 38.17

	Model 1	Model 2	Model 3
(Intercept)	0.01	0.01	0.03
	(0.02)	(0.02)	(0.02)
x	0.99 ***	1.65 ***	0.99 ***
	(0.02)	(0.02)	(0.02)
w		-0.67 ***	-1.01 ***
		(0.01)	(0.01)
z			1.02 ***
			(0.02)
N	10000	10000	10000
R2	0.16	0.39	0.50

*** p < 0.001; ** p < 0.01; * p < 0.05.

38.3.3 Bad Predictive Controls

```
# cleans workspace
rm(list = ls())

# DAG

## specify edges
model <- dagitty("dag{x->y; z->x}")
```

```r
## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, z=1, y=2),
  y = c(x=1, z=2, y=1))

## ggplot
ggdag(model) + theme_dag()
```

```r
n <- 1e4
z <- rnorm(n)
x <- 2 * z + rnorm(n)
y <- x + 2 * rnorm(n)

jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

Similar coefficients, but greater SE when controlling for Z

Another variation is

```r
rm(list = ls())

# DAG

## specify edges
model <- dagitty("dag{x->y; x->z}")


## coordinates for plotting
```

Table 38.18

	Model 1	Model 2
(Intercept)	-0.02	-0.02
	(0.02)	(0.02)
x	0.99 ***	1.00 ***
	(0.01)	(0.02)
z		-0.00
		(0.04)
N	10000	10000
R2	0.55	0.55

*** p < 0.001; ** p < 0.01; * p < 0.05.

```
coordinates(model) <-  list(
  x = c(x=1, z=1, y=2),
  y = c(x=1, z=2, y=1))

## ggplot
ggdag(model) + theme_dag()
```

```
set.seed(1)
n <- 1e4
x <- rnorm(n)
z <- 2 * x + rnorm(n)
y <- x + 2 * rnorm(n)

jtools::export_summs(lm(y ~ x), lm(y ~ x + z))
```

Worse SE when controlling for Z $(0.02 < 0.05)$

38.3.4 Bad Selection Bias

```
# cleans workspace
rm(list = ls())

# DAG

## specify edges
model <- dagitty("dag{x->y; x->z}")


## coordinates for plotting
coordinates(model) <-  list(
  x = c(x=1, z=2, y=2),
  y = c(x=1, z=2, y=1))

## ggplot
```

Table 38.19

	Model 1	Model 2
(Intercept)	0.02	0.02
	(0.02)	(0.02)
x	1.00 ***	0.99 ***
	(0.02)	(0.05)
z		0.00
		(0.02)
N	10000	10000
R2	0.20	0.20

*** p < 0.001; ** p < 0.01; * p < 0.05.

```
ggdag(model) + theme_dag()
```

Not all post-treatment variables are bad.

Controlling for Z is neutral, but it might hurt the precision of the causal effect.

```
library(pcalg)
library(dagitty)
library(causaleffect)
```

By providing a causal diagram, deciding the appropriateness of controls are automated.

- Fusion

- DAGitty

Guide on how to choose confounders: VanderWeele (2019)

In cases where it's hard to determine the plausibility of controls, we might need to further analysis.

`sensemakr` provides such tools.

```
library(sensemakr)
```

In simple cases, we can follow the simple rules of thumb provided by Steinmetz and Block (2022) (p. 614, Fig 2)

Fig. 2 Decision process underlying the consideration of auxiliary variables

Chapter 39

Directed Acyclic Graph

Native R:

- `dagitty`
- `ggdag`
- `dagR`
- `r-causal`: by Center for Causal Discovery. Also available in Python

Publication-ready (with `R` and `Latex`): shinyDAG

Standalone program: DAG program by Sven Knuppel

39.1 Basic Notations

Basic building blocks of DAG

- Mediators (chains): $X \to Z \to Y$
  - controlling for Z blocks (closes) the causal impact of $X \to Y$
- Common causes (forks): $X \leftarrow Z \to Y$
  - Z (i.e., confounder) is a common cause in which it induces a non-causal association between X and Y.
  - Controlling for Z should close this association.
  - Z d-separates X from Y when it blocks (closes) all paths from X to Y (i.e., $X \perp Y|Z$). This applies to both common causes and mediators.
- Common effects (colliders): $X \to Z \leftarrow Y$
  - Not controlling for Z does not induce an association between X and Y

  - Controlling for Z induces a non-causal association between X and Y

Notes:

- A descendant of a variable behavior similarly to that variable (e.g., a descendant of Z can behave like Z and partially control for Z)
- Rule of thumb for multiple Controls: o have Causal inference $X \to Y$, we must
  - Close all backdoor path between X and Y (to eliminate spurious correlation)
  - Do not close any causal path between X and Y (any mediators).

V. MISCELLANEOUS

Chapter 40

Report

Structure

- Exploratory analysis
  - plots
  - preliminary results
  - interesting structure/features in the data
  - outliers
- Model
  - Assumptions
  - Why this model/ How is this model the best one?
  - Consideration: interactions, collinearity, dependence
- Model Fit
  - How well does it fit?
  - Are the model assumptions met?
    - Residual analysis
- Inference/ Prediction
  - Are there different way to support your inference?
- Conclusion
  - Recommendation
  - Limitation of the analysis
  - How to correct those in the future

This chapter is based on the `jtools` package. More information can be found here.

40.1 One summary table

Packages for reporting:

Summary Statistics Table:

- qwraps2
- vtable
- gtsummary
- apaTables
- stargazer

Regression Table

- gtsummary
- sjPlot,sjmisc, sjlabelled
- stargazer: recommended (Example)
- modelsummary

```
library(jtools)
data(movies)
fit <- lm(metascore ~ budget + us_gross + year, data = movies)
summ(fit)
```

Observations	831 (10 missing obs. deleted)
Dependent variable	metascore
Type	OLS linear regression

F(3,827)	26.23
R^2	0.09
Adj. R^2	0.08

	Est.	S.E.	t val.	p
(Intercept)	52.06	139.67	0.37	0.71
budget	-0.00	0.00	-5.89	0.00
us_gross	0.00	0.00	7.61	0.00
year	0.01	0.07	0.08	0.94

Standard errors: OLS

```
summ(
    fit,
    scale = TRUE,
    vifs = TRUE,
    part.corr = TRUE,
    confint = TRUE,
```

```
    pvals = FALSE
) # notice that scale here is TRUE
```

Observations	831 (10 missing obs. deleted)
Dependent variable	metascore
Type	OLS linear regression

F(3,827)	26.23
R^2	0.09
Adj. R^2	0.08

	Est.	2.5%	97.5%	t val.	VIF	partial.r	part.r
(Intercept)	63.01	61.91	64.11	112.23	NA	NA	NA
budget	-3.78	-5.05	-2.52	-5.89	1.31	-0.20	-0.20
us_gross	5.28	3.92	6.64	7.61	1.52	0.26	0.25
year	0.05	-1.18	1.28	0.08	1.24	0.00	0.00

Standard errors: OLS; Continuous predictors are mean-centered and scaled by 1 s.d. The outcome variable remains in its original units.

```
#obtain clsuter-robust SE
data("PetersenCL", package = "sandwich")
fit2 <- lm(y ~ x, data = PetersenCL)
summ(fit2, robust = "HC3", cluster = "firm")
```

Observations	5000
Dependent variable	y
Type	OLS linear regression

F(1,4998)	1310.74
R^2	0.21
Adj. R^2	0.21

	Est.	S.E.	t val.	p
(Intercept)	0.03	0.07	0.44	0.66
x	1.03	0.05	20.36	0.00

Standard errors: Cluster-robust, type = HC3

Model to Equation

```r
# install.packages("equatiomatic") # not available for R 4.2
fit <- lm(metascore ~ budget + us_gross + year, data = movies)
# show the theoretical model
equatiomatic::extract_eq(fit)
# display the actual coefficients
equatiomatic::extract_eq(fit, use_coefs = TRUE)
```

40.2 Model Comparison

```r
fit <- lm(metascore ~ log(budget), data = movies)
fit_b <- lm(metascore ~ log(budget) + log(us_gross), data = movies)
fit_c <- lm(metascore ~ log(budget) + log(us_gross) + runtime, data = movies)
coef_names <- c("Budget" = "log(budget)", "US Gross" = "log(us_gross)",
                "Runtime (Hours)" = "runtime", "Constant" = "(Intercept)")
export_summs(fit, fit_b, fit_c, robust = "HC3", coefs = coef_names)
```

Table 40.1

	Model 1	Model 2	Model 3
Budget	-2.43 ***	-5.16 ***	-6.70 ***
	(0.44)	(0.62)	(0.67)
US Gross		3.96 ***	3.85 ***
		(0.51)	(0.48)
Runtime (Hours)			14.29 ***
			(1.63)
Constant	105.29 ***	81.84 ***	83.35 ***
	(7.65)	(8.66)	(8.82)
N	831	831	831
R2	0.03	0.09	0.17

Standard errors are heteroskedasticity robust. *** p < 0.001; ** p < 0.01; * p < 0.05.

Another package is `modelsummary`

	(1)	(2)	(3)
(Intercept)	38.752	38.752	38.752
	(1.787)	(2.286)	(2.177)
wt	−3.167	−3.167	−3.167
	(0.741)	(0.833)	(0.819)
hp	−0.018	−0.018	−0.018
	(0.012)	(0.010)	(0.013)
cyl	−0.942	−0.942	−0.942
	(0.551)	(0.573)	(0.572)
Num.Obs.	32	32	32
R2	0.843	0.843	0.843
R2 Adj.	0.826	0.826	0.826
AIC	155.5	155.5	155.5
BIC	162.8	162.8	162.8
Log.Lik.	−72.738	−72.738	−72.738
F	50.171	31.065	32.623
RMSE	2.35	2.35	2.35
Std.Errors	IID	HC3	HC4

```r
library(modelsummary)
lm_mod <- lm(mpg ~ wt + hp + cyl, mtcars)
msummary(lm_mod, vcov = c("iid","robust","HC4"))
```

```r
modelplot(lm_mod, vcov = c("iid","robust","HC4"))
```

Another package is `stargazer`

```
library("stargazer")
stargazer(attitude)
#>
#> % Table created by stargazer v.5.2.3 by Marek Hlavac, Social Policy Institute. E-ma
#> % Date and time: Sat, Feb 08, 2025 - 6:54:22 PM
#> \begin{table}[!htbp] \centering
#>   \caption{}
#>   \label{}
#> \begin{tabular}{@{\extracolsep{5pt}}lccccc}
#> \\[-1.8ex]\hline
#> \hline \\[-1.8ex]
#> Statistic & \multicolumn{1}{c}{N} & \multicolumn{1}{c}{Mean} & \multicolumn{1}{c}{S
#> \hline \\[-1.8ex]
#> rating & 30 & 64.633 & 12.173 & 40 & 85 \\
#> complaints & 30 & 66.600 & 13.315 & 37 & 90 \\
#> privileges & 30 & 53.133 & 12.235 & 30 & 83 \\
#> learning & 30 & 56.367 & 11.737 & 34 & 75 \\
#> raises & 30 & 64.633 & 10.397 & 43 & 88 \\
#> critical & 30 & 74.767 & 9.895 & 49 & 92 \\
#> advance & 30 & 42.933 & 10.289 & 25 & 72 \\
#> \hline \\[-1.8ex]
#> \end{tabular}
#> \end{table}
## 2 OLS models
linear.1 <-
    lm(rating ~ complaints + privileges + learning + raises + critical,
       data = attitude)
linear.2 <-
    lm(rating ~ complaints + privileges + learning, data = attitude)
## create an indicator dependent variable, and run a probit model
attitude$high.rating <- (attitude$rating > 70)

probit.model <-
    glm(
        high.rating ~ learning + critical + advance,
        data = attitude,
        family = binomial(link = "probit")
    )
stargazer(linear.1,
          linear.2,
          probit.model,
          title = "Results",
          align = TRUE)
#>
```

```
#> % Table created by stargazer v.5.2.3 by Marek Hlavac, Social Policy Institute. E-mail: marek.h
#> % Date and time: Sat, Feb 08, 2025 - 6:54:22 PM
#> % Requires LaTeX packages: dcolumn
#> \begin{table}[!htbp] \centering
#>   \caption{Results}
#>   \label{}
#> \begin{tabular}{@{\extracolsep{5pt}}lD{.}{.}{-3} D{.}{.}{-3} D{.}{.}{-3} }
#> \\[-1.8ex]\hline
#> \hline \\[-1.8ex]
#>  & \multicolumn{3}{c}{\textit{Dependent variable:}} \\
#> \cline{2-4}
#> \\[-1.8ex] & \multicolumn{2}{c}{rating} & \multicolumn{1}{c}{high.rating} \\
#> \\[-1.8ex] & \multicolumn{2}{c}{\textit{OLS}} & \multicolumn{1}{c}{\textit{probit}} \\
#> \\[-1.8ex] & \multicolumn{1}{c}{(1)} & \multicolumn{1}{c}{(2)} & \multicolumn{1}{c}{(3)}\\
#> \hline \\[-1.8ex]
#>  complaints & 0.692^{***} & 0.682^{***} &  \\
#>   & (0.149) & (0.129) &  \\
#>   & & & \\
#>  privileges & -0.104 & -0.103 &  \\
#>   & (0.135) & (0.129) &  \\
#>   & & & \\
#>  learning & 0.249 & 0.238^{*} & 0.164^{***} \\
#>   & (0.160) & (0.139) & (0.053) \\
#>   & & & \\
#>  raises & -0.033 &  &  \\
#>   & (0.202) &  &  \\
#>   & & & \\
#>  critical & 0.015 &  & -0.001 \\
#>   & (0.147) &  & (0.044) \\
#>   & & & \\
#>  advance &  &  & -0.062 \\
#>   &  &  & (0.042) \\
#>   & & & \\
#>  Constant & 11.011 & 11.258 & -7.476^{**} \\
#>   & (11.704) & (7.318) & (3.570) \\
#>   & & & \\
#> \hline \\[-1.8ex]
#> Observations & \multicolumn{1}{c}{30} & \multicolumn{1}{c}{30} & \multicolumn{1}{c}{30} \\
#> R$^{2}$ & \multicolumn{1}{c}{0.715} & \multicolumn{1}{c}{0.715} &  \\
#> Adjusted R$^{2}$ & \multicolumn{1}{c}{0.656} & \multicolumn{1}{c}{0.682} &  \\
#> Log Likelihood &  &  & \multicolumn{1}{c}{-9.087} \\
#> Akaike Inf. Crit. &  &  & \multicolumn{1}{c}{26.175} \\
#> Residual Std. Error & \multicolumn{1}{c}{7.139 (df = 24)} & \multicolumn{1}{c}{6.863 (df = 26
#> F Statistic & \multicolumn{1}{c}{12.063$^{***}$ (df = 5; 24)} & \multicolumn{1}{c}{21.743$^{**
#> \hline
```

```
#> \hline \\[-1.8ex]
#> \textit{Note:}  & \multicolumn{3}{r}{$^{*}$p$<$0.1; $^{**}$p$<$0.05; $^{***}$p$<$0.
#> \end{tabular}
#> \end{table}
```

```
# Latex
stargazer(
    linear.1,
    linear.2,
    probit.model,
    title = "Regression Results",
    align = TRUE,
    dep.var.labels = c("Overall Rating", "High Rating"),
    covariate.labels = c(
        "Handling of Complaints",
        "No Special Privileges",
        "Opportunity to Learn",
        "Performance-Based Raises",
        "Too Critical",
        "Advancement"
    ),
    omit.stat = c("LL", "ser", "f"),
    no.space = TRUE
)
```

```
# ASCII text output
stargazer(
    linear.1,
    linear.2,
    type = "text",
    title = "Regression Results",
    dep.var.labels = c("Overall Rating", "High Rating"),
    covariate.labels = c(
        "Handling of Complaints",
        "No Special Privileges",
        "Opportunity to Learn",
        "Performance-Based Raises",
        "Too Critical",
        "Advancement"
    ),
    omit.stat = c("LL", "ser", "f"),
    ci = TRUE,
    ci.level = 0.90,
    single.row = TRUE
)
#>
```

```
#> Regression Results
#> ==========================================================================
#>                                          Dependent variable:
#>                            ---------------------------------------------
#>                                            Overall Rating
#>                                     (1)                      (2)
#> --------------------------------------------------------------------------
#> Handling of Complaints   0.692*** (0.447, 0.937) 0.682*** (0.470, 0.894)
#> No Special Privileges    -0.104 (-0.325, 0.118)  -0.103 (-0.316, 0.109)
#> Opportunity to Learn      0.249 (-0.013, 0.512)   0.238* (0.009, 0.467)
#> Performance-Based Raises -0.033 (-0.366, 0.299)
#> Too Critical              0.015 (-0.227, 0.258)
#> Advancement              11.011 (-8.240, 30.262) 11.258 (-0.779, 23.296)
#> --------------------------------------------------------------------------
#> Observations                        30                       30
#> R2                                 0.715                    0.715
#> Adjusted R2                        0.656                    0.682
#> ==========================================================================
#> Note:                                          *p<0.1; **p<0.05; ***p<0.01
```

```r
stargazer(
    linear.1,
    linear.2,
    probit.model,
    title = "Regression Results",
    align = TRUE,
    dep.var.labels = c("Overall Rating", "High Rating"),
    covariate.labels = c(
        "Handling of Complaints",
        "No Special Privileges",
        "Opportunity to Learn",
        "Performance-Based Raises",
        "Too Critical",
        "Advancement"
    ),
    omit.stat = c("LL", "ser", "f"),
    no.space = TRUE
)
```

Correlation Table

```r
correlation.matrix <-
    cor(attitude[, c("rating", "complaints", "privileges")])
stargazer(correlation.matrix, title = "Correlation Matrix")
```

40.3 Changes in an estimate

```
plot_summs(
    fit_c,
    robust = "HC3",
    coefs = coef_names,
    plot.distributions = TRUE
)
```

40.4 Standard Errors

`sandwich` vignette

Type	Applicable	Usage	Reference
`const`		Assume constant variances	
`HC`	`vcovCL`	Heterogeneity	(White, 1980)
`HC0`		White's estimator All other heterogeneity SE methods are derivatives of this. No small sample bias adjustment	
`HC1`	`vcovCL`	Uses a degrees of freedom-based correction When the number of clusters is small, `HC2` and `HC3` are better (Cameron et al., 2008)	(MacKinnon and White, 1985)
`HC2`	`vcovCL`	Better with the linear model, but still applicable for Generalized Linear Models Needs a hat (weighted) matrix	
`HC3`	`vcovCL`	Better with the linear model, but still applicable for Generalized Linear Models Needs a hat (weighted) matrix	
`HC4`	`vcovHC`		(Cribari-Neto, 2004)
`HC4m`	`vcovHC`		(Cribari-Neto et al., 2007)

Type	Applicable Usage	Reference
HC5	vcovHC	(Cribari-Neto and da Silva, 2011)

```
data(cars)
model <- lm(speed ~ dist, data = cars)
summary(model)
#>
#> Call:
#> lm(formula = speed ~ dist, data = cars)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -7.5293 -2.1550  0.3615  2.4377  6.4179
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  8.28391    0.87438   9.474 1.44e-12 ***
#> dist         0.16557    0.01749   9.464 1.49e-12 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 3.156 on 48 degrees of freedom
#> Multiple R-squared:  0.6511, Adjusted R-squared:  0.6438
#> F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
lmtest::coeftest(model, vcov. = sandwich::vcovHC(model, type = "HC1"))
#>
#> t test of coefficients:
#>
#>             Estimate Std. Error t value  Pr(>|t|)
#> (Intercept) 8.283906   0.891860  9.2883 2.682e-12 ***
#> dist        0.165568   0.019402  8.5335 3.482e-11 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

40.5 Coefficient Uncertainty and Distribution

The `ggdist` allows us to visualize uncertainty under both frequentist and Bayesian frameworks

```
library(ggdist)
```

40.6 Descriptive Tables

Export APA theme

```
data("mtcars")

library(flextable)
theme_apa(flextable(mtcars[1:5,1:5]))
```

Export to Latex

```
print(xtable::xtable(mtcars, type = "latex"),
      file = file.path(getwd(), "output", "mtcars_xtable.tex"))

# American Economic Review style
stargazer::stargazer(
    mtcars,
    title = "Testing",
    style = "aer",
    out = file.path(getwd(), "output", "mtcars_stargazer.tex")
)

# other styles include
# Administrative Science Quarterly
# Quarterly Journal of Economics
```

However, the above codes do not play well with notes. Hence, I create my own custom code that follows the AMA guidelines

```
ama_tbl <- function(data, caption, label, note, output_path) {
  library(tidyverse)
  library(xtable)
  # Function to determine column alignment
  get_column_alignment <- function(data) {
    # Start with the alignment for the header row
    alignment <- c("l", "l")

    # Check each column
    for (col in seq_len(ncol(data))[-1]) {
      if (is.numeric(data[[col]])) {
        alignment <- c(alignment, "r")  # Right alignment for numbers
      } else {
        alignment <- c(alignment, "c")  # Center alignment for other data
      }
    }

    return(alignment)
  }
```

```
  data %>%
    # bold + left align first column
    rename_with(~paste("\\multicolumn{1}{l}{\\textbf{", ., "}}"), 1) %>%
    # bold + center align all other columns
    `colnames<-`(ifelse(colnames(.) != colnames(.)[1],
                        paste("\\multicolumn{1}{c}{\\textbf{", colnames(.), "}}"),
                        colnames(.))) %>%

    xtable(caption = caption,
           label = label,
           align = get_column_alignment(data),
           auto = TRUE) %>%
    print(
      include.rownames = FALSE,
      caption.placement = "top",

      hline.after=c(-1, 0),

       # p{0.9\linewidth} sets the width of the column to 90% of the line width, and t

      add.to.row = list(pos = list(nrow(data)), # Add at the bottom of the table
                        command = c(paste0("\\hline \n \\multicolumn{",ncol(data), "}{

      # make sure your heading is untouched (because you manually change it above)
      sanitize.colnames.function = identity,

      # place a the top of the page
      table.placement = "h",

      file = output_path
    )
}
```

```
ama_tbl(
    mtcars,
    caption     = "This is caption",
    label       = "tab:this_is_label",
    note        = "this is note",
    output_path = file.path(getwd(), "output", "mtcars_custom_ama.tex")
)
```

40.7 Visualizations and Plots

You can customize your plots based on your preferred journals. Here, I am creating a custom setting for the American Marketing Association.

American-Marketing-Association-ready theme for plots

```
library(ggplot2)

# check available fonts
# windowsFonts()

# for Times New Roman
# names(windowsFonts()[windowsFonts()=="TT Times New Roman"])
```

```
# Making a theme
amatheme = theme_bw(base_size = 14, base_family = "serif") + # This is Time New Roman

    theme(
        # remove major gridlines
        panel.grid.major   = element_blank(),

        # remove minor gridlines
        panel.grid.minor   = element_blank(),

        # remove panel border
        panel.border       = element_blank(),

        line               = element_line(),

        # change font
        text               = element_text(),

        # if you want to remove legend title
        # legend.title     = element_blank(),

        legend.title       = element_text(size = rel(0.6), face = "bold"),

        # change font size of legend
        legend.text        = element_text(size = rel(0.6)),

        legend.background  = element_rect(color = "black"),

        # legend.margin    = margin(t = 5, l = 5, r = 5, b = 5),
        # legend.key       = element_rect(color = NA, fill = NA),

        # change font size of main title
```

```
        plot.title          = element_text(
            size            = rel(1.2),
            face            = "bold",
            hjust           = 0.5,
            margin          = margin(b = 15)
        ),

        plot.margin         = unit(c(1, 1, 1, 1), "cm"),

        # add black line along axes
        axis.line           = element_line(colour = "black", linewidth = .8),

        axis.ticks          = element_line(),


        # axis title
        axis.title.x        = element_text(size = rel(1.2), face = "bold"),
        axis.title.y        = element_text(size = rel(1.2), face = "bold"),

        # axis text size
        axis.text.y         = element_text(size = rel(1)),
        axis.text.x         = element_text(size = rel(1))
    )
```

Example

```
library(tidyverse)
library(ggsci)
data("mtcars")
yourplot <- mtcars %>%
    select(mpg, cyl, gear) %>%
    ggplot(., aes(x = mpg, y = cyl, fill = gear)) +
    geom_point() +
    labs(title="Some Plot")

yourplot +
    amatheme +
    # choose different color theme
    scale_color_npg()
```

```
yourplot +
    amatheme +
    scale_color_continuous()
```

Other pre-specified themes

```
library(ggthemes)
```

```
# Stata theme
yourplot +
    theme_stata()
```

```
# The economist theme
yourplot +
    theme_economist()
```

```
yourplot +
    theme_economist_white()
```

```
# Wall street journal theme
yourplot +
    theme_wsj()
```

```r
# APA theme
yourplot +
    jtools::theme_apa(
        legend.font.size = 24,
        x.font.size = 20,
        y.font.size = 20
    )
```

Chapter 41

Exploratory Data Analysis

```r
# load to get txhousing data
library(ggplot2)
```

Data Report

Feature Engineering

Missing Data

```r
# install.packages("DataExplorer")
library(DataExplorer)

# creat a html file that contain all reports
create_report(txhousing)

introduce() # see basic info


dummify() # create binary columns from discrete variables
split_columns() # split data into discrete and continuous parts



plot_correlation() # heatmap for discrete var
plot_intro()

plot_missing() # plot missing value
profile_missing() # profile missing values
```

```r
plot_prcomp() # plot PCA
```

Error Identification

```r
# install.packages("dataReporter")
library(dataReporter)
makeDataReport() # detailed report like DataExplorer
```

Summary statistics

```r
library(skimr)
skim() # give only few quick summary stat, not as detailed as the other two packages
```

Not so code-y process

Quick and dirty way to look at your data

```r
# install.packages("rpivotTable")
library(rpivotTable)
# give set up just like Excel table
data %>%
    rpivotTable::rpivotTable()
```

Code generation and wrangling

Shiny-app based Tableu style

```r
# install.packages("esquisse")
library(esquisse)
esquisse::esquisser()
```

Customized your daily/automatic report

```r
# install.packages("chronicle")
library(chronicle)
```

```r
# install.packages("dlookr")
# install.packages("descriptr")
```

Chapter 42

Sensitivity Analysis/ Robustness Check

42.1 Specification curve

- also known as Specification robustness graph or coefficient stability plot

Resources

- In Stata or speccurve
- (Simonsohn et al., 2020)

42.1.1 starbility

- Recommend

Installation

```
devtools::install_github('https://github.com/AakaashRao/starbility')
library(starbility)
```

Example by the package's author

```
library(tidyverse)
library(starbility)
library(lfe)
data("diamonds")
set.seed(43)
indices = sample(1:nrow(diamonds),
                 replace = F,
                 size = round(nrow(diamonds) / 20))
diamonds = diamonds[indices, ]
```

Plot different combinations of controls

```
# If you want to make the diamond dimensions as base control
base_controls = c(
  'Diamond dimensions' = 'x + y + z' # include all variables under 1 dimension
)


perm_controls = c(
  'Depth' = 'depth',
  'Table width' = 'table'
)

nonperm_fe_controls = c(
  'Clarity FE (granular)' = 'clarity',
  'Clarity FE (binary)' = 'high_clarity'
)

# Adding fixed effects
nonperm_fe_controls = c(
  'Clarity FE (granular)' = 'clarity',
  'Clarity FE (binary)' = 'high_clarity'
)

# Adding instrumental variables
instruments = 'x+y+z'

# clustering and weights
diamonds$sample_weights = runif(n = nrow(diamonds))


# robust standard errors
starb_felm_custom = function(spec, data, rhs, ...) {
  spec = as.formula(spec)
  model = lfe::felm(spec, data=data) %>% broom::tidy()

  row = which(model$term==rhs)
  coef = model[row, 'estimate'] %>% as.numeric()
  se   = model[row, 'std.error'] %>% as.numeric()
  p    = model[row, 'p.value'] %>% as.numeric()

  # 99% confidence interval
  z = qnorm(0.995)
  # one-tailed test
  return(c(coef, p/2, coef+z*se, coef-z*se))
}
```

```r
plots = stability_plot(
    data = diamonds,
    lhs = 'price',
    rhs = 'carat',
    error_geom = 'ribbon', # make the plot more aesthetics
    # error_geom = 'none', # if you don't want ribbon (i.e., error bar)
    model = starb_felm_custom,
    cluster = 'cut',
    weights = 'sample_weights',
    # iv = instruments,
    perm = perm_controls,
    base = base_controls,
    # perm_fe = perm_fe_controls,

    # if you want to include fixed effects sequentially (not all combinations)
    # (e.g., you want to test country or state fixed effect, not both )
    # nonperm_fe = nonperm_fe_controls,
    # fe_always = F,  # if you want to have a model without any Fixed Effects

    # sort "asc", "desc", or by fixed effects: "asc-by-fe" or "desc-by-fe"
    sort = "asc-by-fe",

    # if you have less variables and want more aesthetics
    # control_geom = 'circle',
    # point_size = 2,
    # control_spacing = 0.3,


    # error_alpha = 0.2, # change alpha of the error geom
    # point_size = 1.5, # change the size of the coefficient points
    # control_text_size = 10, # change the size of the control labels
    # coef_ylim = c(-5000, 35000), # change the endpoints of the y-axis
    # trip_top = 3, # change the spacing between the two panels

    rel_height = 0.6
)
plots
```

```
# add comments
# replacement_coef_panel = plots[[1]] +
#   scale_y_reverse() +
#   theme(panel.grid.minor = element_blank()) +
#   geom_vline(xintercept = 41,
#              linetype = 'dashed',
#              alpha = 0.4) +
#   annotate(
#     geom = 'label',
#     x = 52,
#     y = 30000,
#     label = 'What a great\nspecification!',
#     alpha = 0.75
#   )
#
# combine_plots(replacement_coef_panel,
#               plots[[2]],
#               rel_height = 0.6)
```

Note:

- $p < 0.01$: red
- $p < 0.05$: green
- $p < 0.1$: blue
- $p > 0.1$: black

More Advanced Stuff

```r
# Step 1: Control Grid

diamonds$high_clarity = diamonds$clarity %in% c('VS1','VVS2','VVS1','IF')

base_controls = c(
  'Diamond dimensions' = 'x + y + z'
)

perm_controls = c(
  'Depth' = 'depth',
  'Table width' = 'table'
)

perm_fe_controls = c(
  'Cut FE' = 'cut',
  'Color FE' = 'color'
)
nonperm_fe_controls = c(
  'Clarity FE (granular)' = 'clarity',
  'Clarity FE (binary)' = 'high_clarity'
)

grid1 = stability_plot(data = diamonds,
                       lhs = 'price',
                       rhs = 'carat',
                       perm = perm_controls,
                       base = base_controls,
                       perm_fe = perm_fe_controls,
                       nonperm_fe = nonperm_fe_controls,
                       run_to=2)

knitr::kable(grid1 %>% head(10))
```

Diamond dimensions	Depth	Table width	Cut FE	Color FE	np_fe
1	0	0	0	0	
1	1	0	0	0	
1	0	1	0	0	
1	1	1	0	0	
1	0	0	1	0	
1	1	0	1	0	
1	0	1	1	0	
1	1	1	1	0	
1	0	0	0	1	
1	1	0	0	1	

```r
# Step 2: Get model expression

grid2 = stability_plot(grid = grid1,
                       data=diamonds,
                       lhs='price',
                       rhs='carat',
                       perm=perm_controls,
                       base=base_controls,
                       run_from=2,
                       run_to=3)


knitr::kable(grid2 %>% head(10))
```

Diamond dimensions	Depth	Table width	np_fe	expr
1	0	0	0	price~carat+x+y+z|0|0|0
1	1	0	0	price~carat+x+y+z+depth|0|0|0
1	0	1	0	price~carat+x+y+z+table|0|0|0
1	1	1	0	price~carat+x+y+z+depth+table|0|0|0
1	0	0	0	price~carat+x+y+z|0|0|0
1	1	0	0	price~carat+x+y+z+depth|0|0|0
1	0	1	0	price~carat+x+y+z+table|0|0|0
1	1	1	0	price~carat+x+y+z+depth+table|0|0|0
1	0	0	0	price~carat+x+y+z|0|0|0
1	1	0	0	price~carat+x+y+z+depth|0|0|0

```r
# Step 3: Estimate models
grid3 = stability_plot(grid = grid2,
                       data=diamonds,
                       lhs='price',
                       rhs='carat',
                       perm=perm_controls,
                       base=base_controls,
                       run_from=3,
                       run_to=4)

knitr::kable(grid3 %>% head(10))
```

Diamond dimensions	Depth	Table width	np_fe	expr	coef	p
1	0	0	0	price~carat+x+y+z|0|0|0	10461.86	p<0
1	1	0	0	price~carat+x+y+z+depth|0|0|0	10808.25	p<0
1	0	1	0	price~carat+x+y+z+table|0|0|0	10423.42	p<0
1	1	1	0	price~carat+x+y+z+depth+table|0|0|0	10851.31	p<0
1	0	0	0	price~carat+x+y+z|0|0|0	10461.86	p<0
1	1	0	0	price~carat+x+y+z+depth|0|0|0	10808.25	p<0
1	0	1	0	price~carat+x+y+z+table|0|0|0	10423.42	p<0
1	1	1	0	price~carat+x+y+z+depth+table|0|0|0	10851.31	p<0
1	0	0	0	price~carat+x+y+z|0|0|0	10461.86	p<0
1	1	0	0	price~carat+x+y+z+depth|0|0|0	10808.25	p<0

```
# Step 4: Get dataframe to draw
dfs = stability_plot(grid = grid3,
                     data=diamonds,
                     lhs='price',
                     rhs='carat',
                     perm=perm_controls,
                     base=base_controls,
                     run_from=4,
                     run_to=5)

coef_grid = dfs[[1]]
control_grid = dfs[[2]]

knitr::kable(coef_grid %>% head(10))
```

Diamond dimensions	Depth	Table width	np_fe	expr	coef	p
1	0	0	0	price~carat+x+y+z|0|0|0	10461.86	p<0
1	1	0	0	price~carat+x+y+z+depth|0|0|0	10808.25	p<0
1	0	1	0	price~carat+x+y+z+table|0|0|0	10423.42	p<0
1	1	1	0	price~carat+x+y+z+depth+table|0|0|0	10851.31	p<0
1	0	0	0	price~carat+x+y+z|0|0|0	10461.86	p<0
1	1	0	0	price~carat+x+y+z+depth|0|0|0	10808.25	p<0
1	0	1	0	price~carat+x+y+z+table|0|0|0	10423.42	p<0
1	1	1	0	price~carat+x+y+z+depth+table|0|0|0	10851.31	p<0
1	0	0	0	price~carat+x+y+z|0|0|0	10461.86	p<0
1	1	0	0	price~carat+x+y+z+depth|0|0|0	10808.25	p<0

```
# Step 5: plot the sensitivity graph
panels = stability_plot(data = diamonds,
                     lhs='price',
                     rhs='carat',
                     coef_grid = coef_grid,
                     control_grid = control_grid,
```

```
                        run_from=5,
                        run_to=6)

stability_plot(data = diamonds,
               lhs='price',
               rhs='carat',
               coef_panel = panels[[1]],
               control_panel = panels[[2]],
               run_from = 6,
               run_to = 7)
```

In step 2, we can modify to use other function (e.g., `glm`)

```
diamonds$above_med_price = as.numeric(diamonds$price > median(diamonds$price))

base_controls = c('Diamond dimensions' = 'x + y + z')

perm_controls = c('Depth' = 'depth',
                  'Table width' = 'table',
                  'Clarity' = 'clarity')
lhs_var = 'above_med_price'
rhs_var = 'carat'

grid1 = stability_plot(
    data = diamonds,
    lhs = lhs_var,
    rhs = rhs_var,
```

```
    perm = perm_controls,
    base = base_controls,
    fe_always = F,
    run_to = 2
)

# Create control part of formula
base_perm = c(base_controls, perm_controls)
grid1$expr = apply(grid1[, 1:length(base_perm)], 1,
                   function(x)
                     paste(base_perm[names(base_perm)[which(x == 1)]],
                           collapse = '+'))

# Complete formula with LHS and RHS variables
grid1$expr = paste(lhs_var, '~', rhs_var, '+', grid1$expr, sep = '')

knitr::kable(grid1 %>% head(10))
```

Diamond dimensions	Depth	Table width	Clarity	np_fe	expr
1	0	0	0		above_med_price~carat+x + y + z
1	1	0	0		above_med_price~carat+x + y + z+depth
1	0	1	0		above_med_price~carat+x + y + z+table
1	1	1	0		above_med_price~carat+x + y + z+depth+tal
1	0	0	1		above_med_price~carat+x + y + z+clarity
1	1	0	1		above_med_price~carat+x + y + z+depth+cla
1	0	1	1		above_med_price~carat+x + y + z+table+clar
1	1	1	1		above_med_price~carat+x + y + z+depth+tal

```
# customer function for the logit model
starb_logit = function(spec, data, rhs, ...) {
  spec = as.formula(spec)
  model = glm(spec, data=data, family='binomial', weights=data$weight) %>%
    broom::tidy()
  row = which(model$term==rhs)
  coef = model[row, 'estimate'] %>% as.numeric()
  se   = model[row, 'std.error'] %>% as.numeric()
  p    = model[row, 'p.value'] %>% as.numeric()

  return(c(coef, p, coef+1.96*se, coef-1.96*se))
}

stability_plot(grid = grid1,
               data = diamonds,
               lhs = lhs_var,
               rhs = rhs_var,
```

```
                    model = starb_logit,
                    perm = perm_controls,
                    base = base_controls,
                    fe_always = F,
                    run_from=3)
```

For getting other specification (e.g., different CI)

```
library(margins)
starb_logit_enhanced = function(spec, data, rhs, ...) {
  # Unpack ...
  l = list(...)
  get_mfx = ifelse(is.null(l$get_mfx), F, T) # Set a default to F

  spec = as.formula(spec)
  if (get_mfx) {
    model = glm(spec, data=data, family='binomial', weights=data$weight) %>%
      margins() %>%
      summary
    row = which(model$factor==rhs)
    coef = model[row, 'AME'] %>% as.numeric()
    se   = model[row, 'SE'] %>% as.numeric()
    p    = model[row, 'p'] %>% as.numeric()
  } else {
    model = glm(spec, data=data, family='binomial', weights=data$weight) %>%
      broom::tidy()
    row = which(model$term==rhs)
```

```
    coef = model[row, 'estimate'] %>% as.numeric()
    se   = model[row, 'std.error'] %>% as.numeric()
    p    = model[row, 'p.value'] %>% as.numeric()
  }

  z = qnorm(0.995)
  return(c(coef, p, coef+z*se, coef-z*se))
}

stability_plot(grid = grid1,
               data = diamonds,
               lhs = lhs_var,
               rhs = rhs_var,
               model = starb_logit_enhanced,
               get_mfx = T,
               perm = perm_controls,
               base = base_controls,
               fe_always = F,
               run_from = 3)
```

To get your customized plot

```
dfs = stability_plot(grid = grid1,
               data = diamonds,
               lhs = lhs_var,
               rhs = rhs_var,
               model = starb_logit_enhanced,
```

```
                get_mfx = T,
                perm = perm_controls,
                base = base_controls,
                fe_always = F,
                run_from = 3,
                run_to = 5)

coef_grid_logit = dfs[[1]]
control_grid_logit = dfs[[2]]

min_space = 0.5

coef_plot = ggplot2::ggplot(coef_grid_logit, aes(
  x = model,
  y = coef,
  shape = p,
  group = p
)) +
  geom_linerange(aes(ymin = error_low, ymax = error_high), alpha = 0.75) +
  geom_point(size = 5, aes(col = p, fill = p), alpha = 1) +
  viridis::scale_color_viridis(discrete = TRUE, option = "D") +
  scale_shape_manual(values = c(15, 17, 18, 19)) +
  theme_classic() +
  geom_hline(yintercept = 0, linetype = 'dotted') +
  ggtitle('A custom coefficient stability plot!') +
  labs(subtitle = "Error bars represent 99% confidence intervals") +
  theme(
    axis.text.x = element_blank(),
    axis.title = element_blank(),
    axis.ticks.x = element_blank()
  ) +
  coord_cartesian(xlim = c(1 - min_space, max(coef_grid_logit$model) + min_space),
                  ylim = c(-0.1, 1.6)) +
  guides(fill = F, shape = F, col = F)


control_plot = ggplot(control_grid_logit) +
  geom_point(aes(x = model, y = y, fill=value), shape=23, size=4) +
  scale_fill_manual(values=c('#FFFFFF', '#000000')) +
  guides(fill=F) +
  scale_y_continuous(breaks = unique(control_grid_logit$y),
                     labels = unique(control_grid_logit$key),
                     limits=c(min(control_grid_logit$y)-1, max(control_grid_logit$y)+1)
  scale_x_continuous(breaks=c(1:max(control_grid_logit$model))) +
  coord_cartesian(xlim=c(1-min_space, max(control_grid_logit$model)+min_space)) +
```

```
  theme_classic() +
  theme(panel.grid.major.y = element_blank(),
        panel.grid.minor.y = element_blank(),
        axis.title = element_blank(),
        axis.text.y = element_text(size=10),
        axis.ticks = element_blank(),
        axis.line = element_blank())

cowplot::plot_grid(coef_plot, control_plot, rel_heights=c(1,0.5),
                   align='v', ncol=1, axis='b')
```

To get different model specification (e.g., probit vs. logit)

```
starb_probit = function(spec, data, rhs, ...) {
    # Unpack ...
    l = list(...)
    get_mfx = ifelse(is.null(l$get_mfx), F, T) # Set a default to F

    spec = as.formula(spec)
    if (get_mfx) {
        model = glm(
            spec,
            data = data,
            family = binomial(link = 'probit'),
            weights = data$weight
        ) %>%
            margins() %>%
```

```
            summary
        row = which(model$factor == rhs)
        coef = model[row, 'AME'] %>% as.numeric()
        se   = model[row, 'SE'] %>% as.numeric()
        p    = model[row, 'p'] %>% as.numeric()
    } else {
        model = glm(
            spec,
            data = data,
            family = binomial(link = 'probit'),
            weights = data$weight
        ) %>%
            broom::tidy()
        row = which(model$term == rhs)
        coef = model[row, 'estimate'] %>% as.numeric()
        se   = model[row, 'std.error'] %>% as.numeric()
        p    = model[row, 'p.value'] %>% as.numeric()
    }

    z = qnorm(0.995)
    return(c(coef, p, coef + z * se, coef - z * se))
}

probit_dfs = stability_plot(
    grid = grid1,
    data = diamonds,
    lhs = lhs_var,
    rhs = rhs_var,
    model = starb_probit,
    get_mfx = T,
    perm = perm_controls,
    base = base_controls,
    fe_always = F,
    run_from = 3,
    run_to = 5
)

# We'll put the probit DFs on the left,
 #so we need to adjust the model numbers accordingly
# so the probit and logit DFs don't plot on top of one another!
coef_grid_probit = probit_dfs[[1]] %>%
    mutate(model = model + max(coef_grid_logit$model))

control_grid_probit = probit_dfs[[2]] %>%
    mutate(model = model + max(control_grid_logit$model))
```

```
coef_grid    = bind_rows(coef_grid_logit, coef_grid_probit)
control_grid = bind_rows(control_grid_logit, control_grid_probit)

panels = stability_plot(
    coef_grid = coef_grid,
    control_grid = control_grid,
    data = diamonds,
    lhs = lhs_var,
    rhs = rhs_var,
    perm = perm_controls,
    base = base_controls,
    fe_always = F,
    run_from = 5,
    run_to = 6
)

coef_plot = panels[[1]] + geom_vline(xintercept = 8.5,
                                     linetype = 'dashed',
                                     alpha = 0.8) +
    annotate(
        geom = 'label',
        x = 4.25,
        y = 1.8,
        label = 'Logit models',
        size = 6,
        fill = '#D3D3D3',
        alpha = 0.7
    ) +
    annotate(
        geom = 'label',
        x = 12.75,
        y = 1.8,
        label = 'Probit models',
        size = 6,
        fill = '#D3D3D3',
        alpha = 0.7
    ) +
    coord_cartesian(ylim = c(-0.5, 1.9))

control_plot = panels[[2]] + geom_vline(xintercept = 8.5,
                                        linetype = 'dashed',
                                        alpha = 0.8)

cowplot::plot_grid(
    coef_plot,
```

```
    control_plot,
    rel_heights = c(1, 0.5),
    align = 'v',
    ncol = 1,
    axis = 'b'
)
```

42.1.2 rdfanalysis

- Not recommend

Installation

```
devtools::install_github("joachim-gassen/rdfanalysis")
```

Example by the package's author

```
library(rdfanalysis)
load(url("https://joachim-gassen.github.io/data/rdf_ests.RData"))
plot_rdf_spec_curve(ests, "est", "lb", "ub")
```

Shiny app for readers to explore

```
design <- define_design(steps = c("read_data",
                                  "select_idvs",
                                  "treat_extreme_obs",
                                  "specify_model",
                                  "est_model"),
                        rel_dir = "vignettes/case_study_code")

shiny_rdf_spec_curve(ests, list("est", "lb", "ub"),
                     design, "vignettes/case_study_code",
                     "https://joachim-gassen.github.io/data/wb_new.csv")
```

42.2 Coefficient stability

(Oster, 2019)

- Coefficient stability can be evident against omitted variable bias.
- But coefficient stability alone can be misleading, but combing with R^2 movement, it can become informative.

Packages

- `mplot`: graphical Model stability and Variable Selection
- `robomit`: Robustness checks for omitted variable bias (implementation of

```r
library(robomit)

# estimate beta
o_beta(
  y     = "mpg",       # dependent variable
  x     = "wt",        # independent treatment variable
  con   = "hp + qsec", # related control variables
  delta = 1,           # delta
  R2max = 0.9,         # maximum R-square
  type  = "lm",        # model type
  data  = mtcars       # dataset
)
#> # A tibble: 10 x 2
#>    Name                            Value
#>    <chr>                           <dbl>
#>  1 beta*                          -2.00
#>  2 (beta*-beta controlled)^2       5.56
#>  3 Alternative Solution 1         -7.01
#>  4 (beta[AS1]-beta controlled)^2   7.05
#>  5 Uncontrolled Coefficient       -5.34
#>  6 Controlled Coefficient         -4.36
#>  7 Uncontrolled R-square           0.753
#>  8 Controlled R-square             0.835
#>  9 Max R-square                    0.9
#> 10 delta                           1
```

42.3 Omitted Variable Bias Quantification

To quantify the bias needed to change the substantive conclusion from a causal inference study.

```r
library(konfound)
pkonfound(
    est_eff = 5,
    std_err = 2,
    n_obs = 1000,
    n_covariates = 5
)
#> Robustness of Inference to Replacement (RIR):
#> To invalidate an inference,  21.506 % of the estimate would have to be due to bias.
#> This is based on a threshold of 3.925 for statistical significance (alpha = 0.05).
#>
#> To invalidate an inference,  215  observations would have to be replaced with cases
#> for which the effect is 0 (RIR = 215).
#>
```

```
#> See Frank et al. (2013) for a description of the method.
#>
#> Citation: Frank, K.A., Maroulis, S., Duong, M., and Kelcey, B. (2013).
#> What would it take to change an inference?
#> Using Rubin's causal model to interpret the
#>         robustness of causal inferences.
#> Education, Evaluation and
#>                        Policy Analysis, 35 437-460.

pkonfound(
    est_eff = 5,
    std_err = 2,
    n_obs = 1000,
    n_covariates = 5,
    to_return = "thresh_plot"
)
```

```
pkonfound(
    est_eff = 5,
    std_err = 2,
    n_obs = 1000,
    n_covariates = 5,
    to_return = "corr_plot"
)
```

To invalidate an inference
Predictor of Interest
Outcome
Rx.cv | Z =
0.134
Rx.cv | Z * Ry.cv | Z =
0.018
Ry.cv | Z =
0.134
Confounding
Variable

Chapter 43

Replication and Synthetic Data

Access to comprehensive data is pivotal for replication, especially in the realm of social sciences. Yet, data are often inaccessible due to proprietary restrictions, privacy concerns, or logistical constraints, making replication a challenge (King, 1995). This chapter explores the nuances of replication, exceptions to its norms, and the significance of synthetic data as a solution.

43.1 The Replication Standard

Replicability in research ensures:

- **Credibility** – Reinforces trust in empirical studies by allowing independent verification.
- **Continuity** – Enables future research to build upon prior findings, promoting cumulative knowledge.
- **Visibility** – Increases readership and citations, benefiting both individual researchers and the broader academic community.

For research to be replicable, adhering to the **replication standard** is essential. This standard requires researchers to provide all necessary information—data, code, and methodological details—so that third parties can independently reproduce the study's findings. While quantitative research often allows for clearer replication, qualitative studies pose challenges due to their depth, contextual nature, and reliance on subjective interpretation.

43.1.1 Solutions for Empirical Replication

Several approaches help address replication challenges in empirical research:

1. **Role of Individual Authors**
   - Researchers must commit to transparency and provide well-documented data and code.
   - Repositories such as the **Inter-University Consortium for Political and Social Research (ICPSR)** offer secure, long-term storage for replication datasets.
2. **Creation of a Replication Data Set**
   - A dedicated **replication dataset** should include original data, relevant supplementary data, and the exact procedures used for analysis.
   - Metadata and documentation should be provided to ensure clarity.
3. **Professional Data Archives**
   - Organizations like **ICPSR, Dataverse, and Zenodo** facilitate open access to datasets while maintaining proper governance over sensitive information.
   - These archives help address data accessibility and preservation issues.
4. **Educational Implications**
   - Teaching replication strengthens students' understanding of empirical methods and reproducibility.
   - Many graduate programs now incorporate replication studies into coursework, emphasizing their importance in methodological rigor.

43.1.2 Free Data Repositories

1. **Zenodo**: Hosted by CERN, it provides a place for researchers to deposit datasets. It's not subject-specific, so it caters to various disciplines.
2. **figshare**: Allows researchers to upload, share, and cite their datasets.
3. **Dryad**: Primarily for datasets associated with published articles in the biological and medical sciences.
4. **OpenICPSR**: A public-facing version of the Inter-University Consortium for Political and Social Research (ICPSR) where researchers can deposit data without any cost.
5. **Harvard Dataverse**: Hosted by Harvard University, this is an open-source repository software application dedicated to archiving, sharing, and citing research data.
6. **Mendeley Data**: A multidisciplinary, free-to-use open access data repository where researchers can upload and share their datasets.
7. **Open Science Framework (OSF)**: Offers both a platform for conducting research and a place to deposit datasets.
8. **PubMed Central**: Specific to life sciences, but it's an open repository for journal articles, preprints, and datasets.
9. **Registry of Research Data Repositories (re3data)**: While not a repository itself, it provides a global registry of research data repositories

from various academic disciplines.

10. **SocArXiv**: An open archive for the social sciences.
11. **EarthArXiv**: A preprints archive for earth science.
12. **Protein Data Bank (PDB)**: For 3D structures of large biological molecules.
13. **Gene Expression Omnibus (GEO)**: A public functional genomics data repository.
14. **The Language Archive (TLA)**: Dedicated to data on languages worldwide, especially endangered languages.
15. **B2SHARE**: A platform for storing and sharing research data sets in various disciplines, especially from European research projects.

43.1.3 Exceptions to Replication

While the replication standard is fundamental to scientific integrity, certain constraints may prevent full adherence. Some common exceptions include:

1. **Confidentiality**
   - Some datasets contain highly sensitive information (e.g., medical records, personal financial data) that cannot be disclosed, even in a fragmented form.
   - Anonymization techniques and data aggregation can sometimes mitigate these concerns, but privacy regulations (e.g., GDPR, HIPAA) impose strict limitations.
2. **Proprietary Data**
   - Datasets owned by corporations, governments, or third-party vendors often have restricted access due to intellectual property concerns.
   - In many cases, researchers can share **summary statistics, derived variables, or synthetic versions** of the data while respecting proprietary restrictions.
3. **Rights of First Publication**
   - Some studies involve data embargoes, where researchers must delay public release until initial publications are completed.
   - Despite embargoes, the essential data and methodology should eventually be accessible to ensure transparency.

43.1.4 Replication Landscape

Brodeur et al. (2025) finds that while AI-assisted teams improve upon AI-led approaches, human-only teams remain the most effective at detecting major errors and ensuring reproducibility in quantitative social science research.

- Human teams and AI-assisted teams achieved similar reproducibility success rates, both significantly outperforming AI-led teams.

- Human-only teams were 57 percentage points more successful than AI-led teams ($p < 0.001$).
- Error detection: Human teams identified significantly more major errors than AI-assisted teams (0.7 more errors per team, $p = 0.017$) and AI-led teams (1.1 more errors per team, $p < 0.001$).
- AI-assisted teams detected 0.4 more errors per team than AI-led teams ($p = 0.029$) but still fewer than human teams.
- Robustness checks: Both human and AI-assisted teams were significantly better than AI-led teams in proposing (25 percentage points, $p = 0.017$) and implementing (33 percentage points, $p = 0.005$) comprehensive robustness checks.

43.2 Synthetic Data

Synthetic data, which models real data while ensuring anonymity, is becoming an essential tool in research. By generating artificial datasets that retain key statistical properties of the original data, researchers can **preserve privacy, enhance data accessibility, and facilitate replication**. However, synthetic data also introduces complexities and should be used with caution.

43.2.1 Benefits of Synthetic Data

1. **Privacy Preservation**
   - Protects sensitive or proprietary information while enabling research collaboration.
2. **Data Fairness and Augmentation**
   - Helps mitigate biases by generating more balanced datasets.
   - Can supplement real data when sample sizes are limited.
3. **Acceleration in Research**
   - Allows for data sharing in environments where access to real data is restricted.
   - Enables large-scale simulations without legal or ethical constraints.

43.2.2 Concerns and Limitations

1. **Misconceptions About Privacy**
   - Synthetic data does not guarantee absolute privacy—re-identification risks remain if it is too similar to the real dataset.
2. **Challenges with Data Outliers**
   - Rare but important data points may be poorly represented or excluded.

3. **Risks of Solely Relying on Synthetic Data**
   - Models trained exclusively on synthetic data may lack generalizability.
   - Differences between real and synthetic distributions can introduce biases.

43.2.3 Further Insights on Synthetic Data

Synthetic data acts as a bridge between **model-centric and data-centric perspectives**, making it a vital tool in modern research. An analogy can be drawn to **viewing a replica of the Mona Lisa**—the essence remains, but the original is securely stored.

For a deeper dive into synthetic data and its applications, refer to (Jordon et al., 2022).

43.2.4 Generating Synthetic Data

When generating synthetic data, the approach depends on whether researchers have full access to the original dataset or are working under restricted conditions.

43.2.4.1 When You Have Access to the Original Dataset

If researchers can directly use the dataset, various techniques can be employed to generate synthetic data while preserving the statistical properties of the original:

- **Statistical Approaches**
  - **Parametric models (e.g., Gaussian Mixture Models)**
    - Fit statistical distributions to real data and sample synthetic observations.
- **Machine Learning-Based Methods**
  - **Variational Autoencoders (VAEs)** – Useful for structured, complex data representations.
  - **Generative Adversarial Networks (GANs)** – Effective for generating high-dimensional data (e.g., tabular, image, and text data).
  - **CTGAN (Conditional Tabular GAN)** – Specifically designed for structured, tabular datasets, addressing categorical and imbalanced data challenges.
- **Differential Privacy Techniques**
  - **Noise Addition** – Introduces controlled noise while maintaining the overall statistical structure.

43.2.4.2 When You Have a Restricted Dataset

In cases where **data cannot be exported** due to security, privacy, or proprietary constraints, researchers must rely on alternative strategies to generate synthetic data:

- **Summarization and Approximation**
  - Extract summary statistics (e.g., means, variances, correlations) to approximate the dataset's structure.
  - If permitted, share aggregated or anonymized data instead of raw observations.
- **Server-Based Computation**
  - Conduct in-server analyses where raw data remains inaccessible, but synthetic outputs can be generated on the secure system.
- **Synthetic Data Generation with Preserved Properties**
  - Use models trained on the secure dataset to produce synthetic data without directly copying real observations.
  - Ensure that key statistical relationships are maintained, even if individual values differ.

43.3 Application

43.3.1 Original Dataset

1. Import libraries

```r
library(copula)
library(moments)
library(PerformanceAnalytics)  # For correlation plots
library(ggplot2)
library(dplyr)
```

2. Simulate a Complex, Nonlinear, Hierarchical Time Series

Suppose we have:

- $G = 3$ groups (e.g., groups 1, 2, 3)
- Each group has $N = 50$ units (e.g., individuals or devices)
- Each unit is measured at $T = 20$ time points

We'll create four continuous variables, `X1` through `X4`, each influenced by:

- A group-level random effect (different intercept by group)
- A unit-level random effect (different intercept by unit)
- Time (with some nonlinear relationships, e.g., sine, polynomial)
- Nonlinear cross-relationships among X1–X4

This gives us a total of $3 \times 50 \times 20 = 3000$ rows in the "original" dataset.

```r
set.seed(123)  # For reproducibility

G <- 3    # Number of groups
N <- 50   # Units per group
Tt <- 20  # Time points per unit

# Create a data frame structure
df_list <- list()

for(g in 1:G) {

  # Group-level random intercept
  group_intercept <- rnorm(1, mean = 0, sd = 1)

  for(u in 1:N) {

    # Unit-level random intercept
    unit_intercept <- rnorm(1, mean = 0, sd = 0.5)

    # Simulate time points
    time_points <- 1:Tt

    # Create some base patterns
    X1_base <- group_intercept + unit_intercept +
               sin(0.2 * time_points) +  # Nonlinear time pattern
               rnorm(Tt, mean = 0, sd = 0.2)

    # Introduce different relationships for X2, X3, X4
    # Some polynomial in time, plus dependence on X1
    X2_base <- (X1_base^2) + 0.5 * time_points + rnorm(Tt, 0, 0.3)
    X3_base <- 1 + group_intercept - 0.3 * X1_base + log(time_points+1) +
               rnorm(Tt, mean = 0, sd = 0.2)
    X4_base <- exp(0.1 * X1_base) + 0.2 * (X2_base) - 0.5 * (X3_base) +
               rnorm(Tt, mean = 0, sd = 0.5)

    df_temp <- data.frame(
      group = g,
      unit  = paste0("G", g, "_U", u),
      time  = time_points,
      X1    = X1_base,
      X2    = X2_base,
      X3    = X3_base,
      X4    = X4_base
    )
```

```
    df_list[[length(df_list) + 1]] <- df_temp
  }
}

df_original <- do.call(rbind, df_list)
row.names(df_original) <- NULL

# Inspect the first rows
head(df_original)
#>   group  unit time           X1        X2        X3         X4
#> 1     1 G1_U1    1 -0.165153398 0.2194743 0.9291383  0.3963423
#> 2     1 G1_U1    2 -0.272044371 0.8553408 2.0535411 -0.3918278
#> 3     1 G1_U1    3 -0.085064371 1.3197242 2.0929304 -0.3268864
#> 4     1 G1_U1    4  0.384804697 1.6420667 1.7088991  0.6649585
#> 5     1 G1_U1    5  0.258089835 2.8179465 2.0732799  0.7771992
#> 6     1 G1_U1    6  0.003462448 3.0460239 2.2910647  0.4905209
```

3. Explore the Original Dataset

Let's do some **descriptive statistics** and look at the **correlations** among X1–X4.
Because we have repeated measures (time series) nested in units and groups, these correlations are "pooled" across all rows. This is a simplification, but it will let us demonstrate how to do a copula-based synthetic approach.

```
# Descriptive stats (overall)
summary(df_original[, c("X1", "X2", "X3", "X4")])
#>        X1                X2                 X3                 X4
#>  Min.   :-4.8316   Min.   :-0.02119   Min.   :-0.7811   Min.   :-1.2191
#>  1st Qu.:-1.8272   1st Qu.: 4.47120   1st Qu.: 1.5865   1st Qu.: 0.5339
#>  Median :-0.3883   Median : 7.08316   Median : 2.4366   Median : 1.1073
#>  Mean   :-0.6520   Mean   : 7.76411   Mean   : 2.3832   Mean   : 1.3118
#>  3rd Qu.: 0.4869   3rd Qu.: 9.65271   3rd Qu.: 3.3120   3rd Qu.: 1.8859
#>  Max.   : 2.5247   Max.   :33.30223   Max.   : 4.9642   Max.   : 6.8820

# Skewness & Kurtosis
apply(df_original[, c("X1", "X2", "X3", "X4")], 2, skewness)
#>         X1         X2         X3         X4
#> -0.3515255  1.3614289 -0.3142704  0.9435834
apply(df_original[, c("X1", "X2", "X3", "X4")], 2, kurtosis)
#>         X1         X2         X3         X4
#> -0.7689287  2.6280956 -0.6086295  1.1138128

# Correlation matrix
(cor_mat <- cor(df_original[, c("X1", "X2", "X3", "X4")]))
#>            X1         X2         X3         X4
```

```
#> X1  1.0000000 -0.6865655  0.3948406 -0.7035793
#> X2 -0.6865655  1.0000000  0.1301602  0.7532715
#> X3  0.3948406  0.1301602  1.0000000 -0.3660620
#> X4 -0.7035793  0.7532715 -0.3660620  1.0000000

chart.Correlation(df_original[, c("X1", "X2", "X3", "X4")],
                  histogram = TRUE, pch = 19)
```

4. Convert to Pseudo-Observations for Copula Fitting

Copulas need variables in $[0, 1]$ space, so we use the **empirical CDF** ("probability integral transform") on each variable.

> **Important**: We have discrete variables like `group`, or ID-like columns such as `unit`. For the synthetic generation of group/unit/time, we have multiple strategies:
>
> 1. **Model them as random**: e.g., re-sample `group`, `unit`, `time` from the original distribution.
> 2. **Treat them as continuous** in a copula (not recommended for true IDs).
> 3. **Do a hierarchical approach**: fit a separate copula for each group or each time slice (advanced).

For simplicity, we'll:

1. Re-sample `group` and `time` from their original distributions (like "bootstrapping").

2. Use a **multivariate copula** only for `X1-X4`.

```
# Extract the numeric columns we want to transform
original_data <- df_original[, c("X1","X2","X3","X4")]

# Convert each to uniform [0,1] by empirical CDF
u_data <- pobs(as.matrix(original_data))

# Check ranges (should be between 0 and 1)
apply(u_data, 2, range)
#>                X1           X2           X3           X4
#> [1,] 0.0003332223 0.0003332223 0.0003332223 0.0003332223
#> [2,] 0.9996667777 0.9996667777 0.9996667777 0.9996667777
```

5. Fit a Copula Model

We'll fit a **Gaussian copula** (you could try **t-copula** or **vine copulas** for heavier tails or more complex dependencies). We use maximum likelihood estimation:

```
# Define an unstructured Gaussian copula
gaussCop <- normalCopula(dim = ncol(u_data), dispstr = "un")

# Fit to the pseudo-observations
fit_gauss <- fitCopula(gaussCop, data = u_data, method = "ml")

summary(fit_gauss)
#> Call: fitCopula(gaussCop, data = u_data, ... = pairlist(method = "ml"))
#> Fit based on "maximum likelihood" and 3000 4-dimensional observations.
#> Normal copula, dim. d = 4
#>       Estimate Std. Error
#> rho.1  -0.5746      0.010
#> rho.2   0.3024      0.014
#> rho.3  -0.5762      0.010
#> rho.4   0.2188      0.013
#> rho.5   0.6477      0.008
#> rho.6  -0.3562      0.012
#> The maximized loglikelihood is 2885
#> Optimization converged
#> Number of loglikelihood evaluations:
#> function gradient
#>       68       14
```

Check the estimated correlation matrix within the copula. This should reflect the dependency among X1–X4 (though not time, group, or unit).

6. Generate Synthetic Data

- Synthetic X1–X4

1. **Sample** from the fitted copula to get synthetic $[0, 1]$ values.
2. **Invert** them via the original empirical distributions (quantiles).

```
n_synth <- nrow(df_original)  # same size as original

# Sample from the copula
u_synth <- rCopula(n_synth, fit_gauss@copula)

# Convert from [0,1] -> real scale by matching original distribution
synth_X <- data.frame(
    X1_synth = quantile(original_data$X1, probs = u_synth[, 1],
                        type = 8),
    X2_synth = quantile(original_data$X2, probs = u_synth[, 2],
                        type = 8),
    X3_synth = quantile(original_data$X3, probs = u_synth[, 3],
                        type = 8),
    X4_synth = quantile(original_data$X4, probs = u_synth[, 4],
                        type = 8)
)

head(synth_X)
#>                 X1_synth X2_synth  X3_synth   X4_synth
#> 68.93791982%  0.2965446 3.488621 2.0876565  0.9336342
#> 40.75838264% -0.8903883 5.341148 3.4134562  0.4857101
#> 80.51755107%  0.6841892 2.536933 0.5329199  1.3494999
#> 29.87601496% -1.5123574 8.127288 0.4861639  3.2120501
#> 68.75449744%  0.2855353 3.219903 2.1439338 -0.3848253
#> 35.73763642% -1.1651872 5.717179 2.7372298  0.6252287
```

- Synthetic Group, Unit, and Time

A simple approach is to:

- **Re-sample "group"** with the same probabilities as the original distribution.
- **Re-sample "unit"** within each group or treat it as purely random labels (depending on your needs).
- **Re-sample "time"** from the original distribution or replicate the same time points.

Below, we do a simplistic approach: for each row, pick a random row from the original data to copy `group`, `unit`, and `time`. This preserves the real distribution of group/time pairs and the frequency of each unit. (But it does **not** preserve the original time-series ordering or autoregressive structure!)

```
indices <-
    sample(seq_len(nrow(df_original)),
```

```
            size = n_synth,
            replace = TRUE)
synth_meta <- df_original[indices, c("group", "unit", "time")]

# Combine the meta-info with the synthetic X's
df_synth <- cbind(synth_meta, synth_X)

head(df_synth)
#>      group   unit time   X1_synth X2_synth  X3_synth   X4_synth
#> 1029     2  G2_U2    9  0.2965446 3.488621 2.0876565  0.9336342
#> 2279     3 G3_U14   19 -0.8903883 5.341148 3.4134562  0.4857101
#> 1885     2 G2_U45    5  0.6841892 2.536933 0.5329199  1.3494999
#> 2251     3 G3_U13   11 -1.5123574 8.127288 0.4861639  3.2120501
#> 1160     2  G2_U8   20  0.2855353 3.219903 2.1439338 -0.3848253
#> 2222     3 G3_U12    2 -1.1651872 5.717179 2.7372298  0.6252287
```

If you **need** to preserve the exact time-ordering or real "per-unit" correlation across time, you'd need a more advanced approach (e.g., separate copula by unit or a hierarchical time-series model).

7. Validate the Synthetic Data

- Compare Descriptive Statistics

```
# Original
orig_means  <- colMeans(df_original[, c("X1", "X2", "X3", "X4")])
orig_sds    <- apply(df_original[, c("X1", "X2", "X3", "X4")], 2, sd)
orig_skew   <-
    apply(df_original[, c("X1", "X2", "X3", "X4")], 2, skewness)
orig_kurt   <-
    apply(df_original[, c("X1", "X2", "X3", "X4")], 2, kurtosis)

# Synthetic
synth_means <-
    colMeans(df_synth[, c("X1_synth", "X2_synth", "X3_synth", "X4_synth")])
synth_sds   <-
    apply(df_synth[, c("X1_synth", "X2_synth", "X3_synth", "X4_synth")], 2, sd)
synth_skew  <-
    apply(df_synth[, c("X1_synth", "X2_synth", "X3_synth", "X4_synth")], 2, skewness)
synth_kurt  <-
    apply(df_synth[, c("X1_synth", "X2_synth", "X3_synth", "X4_synth")], 2, kurtosis)

cat(
    "### Means ###\nOriginal:",
    round(orig_means, 3),
    "\nSynthetic:",
    round(synth_means, 3),
```

```r
    "\n\n"
)
#> ### Means ###
#> Original: -0.652 7.764 2.383 1.312
#> Synthetic: -0.642 7.741 2.372 1.287

cat(
    "### SDs ###\nOriginal:",
    round(orig_sds, 3),
    "\nSynthetic:",
    round(synth_sds, 3),
    "\n\n"
)
#> ### SDs ###
#> Original: 1.448 4.937 1.16 1.102
#> Synthetic: 1.449 5.046 1.159 1.102

cat(
    "### Skewness ###\nOriginal:",
    round(orig_skew, 3),
    "\nSynthetic:",
    round(synth_skew, 3),
    "\n\n"
)
#> ### Skewness ###
#> Original: -0.352 1.361 -0.314 0.944
#> Synthetic: -0.349 1.353 -0.317 1.017

cat(
    "### Kurtosis ###\nOriginal:",
    round(orig_kurt, 3),
    "\nSynthetic:",
    round(synth_kurt, 3),
    "\n\n"
)
#> ### Kurtosis ###
#> Original: -0.769 2.628 -0.609 1.114
#> Synthetic: -0.744 2.465 -0.619 1.381
```

- Compare Correlation Matrices

```r
cat("Original correlation:\n")
#> Original correlation:
round(cor(df_original[, c("X1", "X2", "X3", "X4")]), 3)
#>        X1     X2     X3     X4
#> X1  1.000 -0.687  0.395 -0.704
```

```
#> X2 -0.687  1.000  0.130  0.753
#> X3  0.395  0.130  1.000 -0.366
#> X4 -0.704  0.753 -0.366  1.000

cat("\nSynthetic correlation:\n")
#>
#> Synthetic correlation:
round(cor(df_synth[, c("X1_synth", "X2_synth", "X3_synth", "X4_synth")]), 3)
#>          X1_synth X2_synth X3_synth X4_synth
#> X1_synth    1.000   -0.562    0.288   -0.565
#> X2_synth   -0.562    1.000    0.195    0.636
#> X3_synth    0.288    0.195    1.000   -0.346
#> X4_synth   -0.565    0.636   -0.346    1.000
```

- Visual Comparison of Distributions

```
par(mfrow = c(2, 2))
vars <- c("X1", "X2", "X3", "X4")
for (i in seq_along(vars)) {
    hist(
        df_original[[vars[i]]],
        probability = TRUE,
        breaks = 30,
        main = paste("Original", vars[i]),
        col = rgb(1, 0, 0, 0.5)
    )
    hist(
        df_synth[[paste0(vars[i], "_synth")]],
        probability = TRUE,
        breaks = 30,
        main = paste("Synthetic", vars[i]),
        col = rgb(0, 0, 1, 0.5),
        add = TRUE
    )
    legend(
        "topright",
        legend = c("Original", "Synthetic"),
        fill = c(rgb(1, 0, 0, 0.5), rgb(0, 0, 1, 0.5))
    )
}
```

- Correlation Plot for Synthetic Data

```
chart.Correlation(df_synth[, c("X1_synth", "X2_synth", "X3_synth", "X4_synth")],
                  histogram = TRUE, pch = 19)
```

- **Indistinguishability**: If the synthetic summary statistics (means, variances, skewness, kurtosis) and correlation structure match closely, the synthetic data is often "indistinguishable" from the original for many analytical purposes.

- **Hierarchical / Time-Series**: True hierarchical time-series replication (i.e., preserving each unit's time autocorrelation and group structure) may require more advanced methods, such as:
  - **Hierarchical copulas** or **vine copulas** over time slices.
  - **Mixed-effects / random-effects** modeling (e.g., for group and unit) plus a copula for residuals.
  - **Deep generative approaches** (e.g., TimeGAN) for strong temporal dynamics, currently more common in Python.
- **Categorical Variables**: For strictly categorical variables (e.g., group, unit ID), you can:
  - Fit separate copulas within each group.
  - Convert categories to numeric in a naive way (not recommended for actual IDs) or use specialized **discrete copulas**.
- **Privacy Considerations**: Even if data is synthetic, do check that it doesn't inadvertently leak private information (e.g., via memorizing outliers). Techniques like **differential privacy** or **post-hoc checks** might be required.

43.3.2 Restricted Dataset

1. **Generate the "Original" Complex Dataset**

We'll simulate a hierarchical time-series with:

- $G = 3$ groups
- $N = 50$ units per group
- $T = 20$ time points per unit
- Nonlinear relationships between `X1`, `X2`, `X3`, `X4`.

```
# Step 1: Generate "df_original" (what the partner owns internally)
set.seed(123)  # For reproducibility

G <- 3    # Number of groups
N <- 50   # Units per group
Tt <- 20  # Time points per unit

df_list <- list()

for(g in 1:G) {

  # Group-level random intercept
  group_intercept <- rnorm(1, mean = 0, sd = 1)
```

```r
  for(u in 1:N) {

    # Unit-level random intercept
    unit_intercept <- rnorm(1, mean = 0, sd = 0.5)

    # Simulate time points
    time_points <- 1:Tt

    # Create some base patterns (X1)
    X1_base <- group_intercept + unit_intercept +
               sin(0.2 * time_points) +  # Nonlinear time pattern
               rnorm(Tt, mean = 0, sd = 0.2)

    # X2 depends on polynomial in time, plus dependence on X1
    X2_base <- (X1_base^2) + 0.5 * time_points + rnorm(Tt, 0, 0.3)

    # X3 depends on group intercept, negative correlation with X1, and log(time)
    X3_base <- 1 + group_intercept - 0.3 * X1_base + log(time_points + 1) +
               rnorm(Tt, mean = 0, sd = 0.2)

    # X4 depends on X1, X2, X3 in a more complex, nonlinear form
    X4_base <- exp(0.1 * X1_base) + 0.2 * X2_base - 0.5 * X3_base +
               rnorm(Tt, mean = 0, sd = 0.5)

    df_temp <- data.frame(
      group = g,
      unit  = paste0("G", g, "_U", u),
      time  = time_points,
      X1    = X1_base,
      X2    = X2_base,
      X3    = X3_base,
      X4    = X4_base
    )

    df_list[[length(df_list) + 1]] <- df_temp
  }
}

df_original <- do.call(rbind, df_list)
row.names(df_original) <- NULL

# Inspect the first rows (just for illustration)
head(df_original)
#>   group  unit time           X1        X2        X3         X4
#> 1     1 G1_U1    1 -0.165153398 0.2194743 0.9291383  0.3963423
```

```
#> 2     1 G1_U1    2 -0.272044371 0.8553408 2.0535411 -0.3918278
#> 3     1 G1_U1    3 -0.085064371 1.3197242 2.0929304 -0.3268864
#> 4     1 G1_U1    4  0.384804697 1.6420667 1.7088991  0.6649585
#> 5     1 G1_U1    5  0.258089835 2.8179465 2.0732799  0.7771992
#> 6     1 G1_U1    6  0.003462448 3.0460239 2.2910647  0.4905209
```

At this point, imagine `df_original` lives *only* on the partner's server and cannot be exported in its raw form.

2. **Manually Collect Summary Statistics (Inside Secure Server)**

Within the secure environment, you would run commands to get:

- Means, standard deviations for each variable
- Correlation matrix
- Group distribution info (how many groups, units, etc.)
- Any other relevant stats (min, max, skewness, kurtosis, etc.) you might use

Below, we'll do that *directly* in code—**but in reality**, you would just write these numbers down or save them in a doc, not export the raw data.

```r
# Step 2: Summaries from "df_original" (pretend we can't take the actual df out)
library(dplyr)

# For demonstration, we'll compute them here:
stats_summary <- df_original %>%
  summarise(
    mean_X1 = mean(X1),
    mean_X2 = mean(X2),
    mean_X3 = mean(X3),
    mean_X4 = mean(X4),

    sd_X1 = sd(X1),
    sd_X2 = sd(X2),
    sd_X3 = sd(X3),
    sd_X4 = sd(X4)
  )

# Extract the correlation matrix among (X1, X2, X3, X4)
cor_matrix <- cor(df_original[, c("X1","X2","X3","X4")])

# Also note the group info
unique_groups <- unique(df_original$group)
group_sizes   <- table(df_original$group)
N_groups      <- length(unique_groups)
```

```
unit_example  <- length(unique(df_original$unit[df_original$group == 1]))
time_points   <- length(unique(df_original$time[df_original$group == 1 & df_original$unit == "G1_

# Print them out as if we wrote them down
stats_summary
#>     mean_X1  mean_X2  mean_X3  mean_X4    sd_X1    sd_X2    sd_X3    sd_X4
#> 1 -0.651988 7.764111 2.383214 1.311789 1.448297 4.937284 1.160375 1.102264
cor_matrix
#>            X1         X2         X3         X4
#> X1  1.0000000 -0.6865655  0.3948406 -0.7035793
#> X2 -0.6865655  1.0000000  0.1301602  0.7532715
#> X3  0.3948406  0.1301602  1.0000000 -0.3660620
#> X4 -0.7035793  0.7532715 -0.3660620  1.0000000
group_sizes
#>
#>    1    2    3
#> 1000 1000 1000
N_groups
#> [1] 3
unit_example
#> [1] 50
time_points
#> [1] 20
```

Example Output (numbers will vary):

- Means, SDs of each variable
- 4×4 correlation matrix
- `group_sizes`: each group has 50×20 = 1000 rows
- `N_groups`: 3

Simulating What We "Take Out"

Pretend these are the **only** data you're allowed to copy into your local machine:

```
# (Pretend these are typed or copy-pasted from the secure environment)

# Means and SDs:
means <- c(stats_summary$mean_X1, stats_summary$mean_X2,
           stats_summary$mean_X3, stats_summary$mean_X4)
sds   <- c(stats_summary$sd_X1,   stats_summary$sd_X2,
           stats_summary$sd_X3,   stats_summary$sd_X4)

# Correlation matrix:
R <- cor_matrix
```

```
# Hierarchical structure info:
G_outside   <- N_groups  # 3
N_outside   <- unit_example  # 50
Tt_outside  <- time_points   # 20
```

4. **Reconstruct Covariance Matrix and Distribution (Outside)**

Outside, you now have:

- A mean vector for `(X1, X2, X3, X4)`
- Standard deviations for each
- A correlation matrix R
- Basic knowledge: 3 groups, 50 units each, 20 time points each (or however the real data is structured)

Build the covariance Σ from the correlation matrix and SDs:

```
# Step 3: Covariance matrix = diag(SDs) %*% R %*% diag(SDs)
Sigma <- diag(sds) %*% R %*% diag(sds)
Sigma
#>            [,1]       [,2]       [,3]       [,4]
#> [1,]  2.0975651 -4.9093928  0.6635565 -1.1231984
#> [2,] -4.9093928 24.3767734  0.7457009  4.0994482
#> [3,]  0.6635565  0.7457009  1.3464702 -0.4682079
#> [4,] -1.1231984  4.0994482 -0.4682079  1.2149866
```

4. **Generate a Synthetic Dataset Matching Those Stats**

We'll replicate the same **hierarchical shape**: 3 groups, 50 units, 20 time points. But we'll fill in `(X1, X2, X3, X4)` by sampling from **multivariate normal** with `(means, Sigma)`.

> In practice, you might want to add back random intercepts for groups or time trends if your manual stats include that. However, if all you have are overall means, SDs, and a correlation matrix, the simplest approach is to assume a single global distribution for X1–X4.

```
library(MASS)

set.seed(999)  # Synthetic data seed (different from original)

df_synth <- data.frame()

for(g in 1:G_outside) {
  for(u in 1:N_outside) {
    for(t in 1:Tt_outside) {
      # Draw one sample from the 4D normal
```

```
      X_vector <- mvrnorm(n = 1, mu = means, Sigma = Sigma)

      df_temp <- data.frame(
        group = g,
        unit  = paste0("G", g, "_U", u),
        time  = t,
        X1    = X_vector[1],
        X2    = X_vector[2],
        X3    = X_vector[3],
        X4    = X_vector[4]
      )
      df_synth <- rbind(df_synth, df_temp)
    }
  }
}

# Check the first rows of the synthetic dataset
head(df_synth)
#>   group  unit time          X1        X2       X3        X4
#> 1     1 G1_U1    1  0.55555434  9.484499 3.729036 1.3156911
#> 2     1 G1_U1    2 -1.33012636  9.275848 3.202148 0.1467821
#> 3     1 G1_U1    3 -0.02332833 12.858857 3.425945 2.1621458
#> 4     1 G1_U1    4  0.84430687  3.253564 1.536901 0.5493251
#> 5     1 G1_U1    5  0.04440973  7.617650 1.552945 1.0185252
#> 6     1 G1_U1    6 -2.70087059 13.626038 1.959473 2.6224373
```

At this point, `df_synth` is a dataset that has the same shape (3 groups $\times$ 50 units $\times$ 20 time points = 3000 rows) and is drawn from the same approximate distribution (matching the partner's means, SDs, correlation matrix).

Alternatively, if the goal is to capture even skewness and kurtosis, it's a bit more complex.

```
# Load required libraries
library(MASS)   # For multivariate normal correlation structure
library(sn)     # For skewed normal distribution

# Step 1: Define Structure
num_groups <- 10   # Number of groups
num_timepoints <- 50  # Time series length per group
total_samples <- num_groups * num_timepoints  # Total data points

# Define statistical properties for each group
set.seed(123)  # For reproducibility

group_means <- rnorm(num_groups, mean=50, sd=10)  # Each group has a different mean
```

```
group_variances <- runif(num_groups, 50, 150)  # Random variance per group
group_skewness <- runif(num_groups, -1, 2)  # Skewness for each group
# group_kurtosis <- runif(num_groups, 3, 6)  # Excess kurtosis for each group

# Define AR(3) autocorrelation coefficients
phi <- c(0.5, 0.3, 0.2)  # AR(3) coefficients (must sum to < 1 for stationarity)
p <- length(phi)  # Order of the AR process

# Define correlation matrix for groups (cross-sectional correlation)
group_corr_matrix <- matrix(0.5, nrow=num_groups, ncol=num_groups)  # Moderate correla
diag(group_corr_matrix) <- 1  # Set diagonal to 1 for perfect self-correlation

# Cholesky decomposition for group-level correlation
chol_decomp <- chol(group_corr_matrix)

# Step 2: Generate Hierarchical Time Series Data
data_list <- list()

for (g in 1:num_groups) {
    # Generate base time series (AR(p) process)
    ts_data <- numeric(num_timepoints)

    # Initialize first 'p' values randomly
    ts_data[1:p] <- rnorm(p, mean=group_means[g], sd=sqrt(group_variances[g]))

    for (t in (p+1):num_timepoints) {
        # AR(p) process with multiple past values
        ts_data[t] <- sum(phi * ts_data[(t-p):(t-1)]) +
            rnorm(1, mean=0, sd=sqrt(group_variances[g] * (1 - sum(phi^2))))
    }

    # Add skewness using skewed normal distribution
    ts_data <- rsn(num_timepoints, xi=mean(ts_data), omega=sd(ts_data), alpha=group_ske

    # Store data in list
    data_list[[g]] <- data.frame(
        Group = g,
        Time = 1:num_timepoints,
        Value = ts_data
    )
}

# Combine all group data into a single DataFrame
df <- do.call(rbind, data_list)
```

```r
# Step 3: Apply Cross-Group Correlation
# Reshape the dataset for correlation application
wide_df <- reshape(df, idvar="Time", timevar="Group", direction="wide")

# Apply correlation across groups at each time step
for (t in 1:num_timepoints) {
    wide_df[t, -1] <- as.numeric(as.matrix(wide_df[t, -1]) %*% chol_decomp)
}

# Convert back to long format correctly
long_df <- reshape(wide_df,
                   varying=colnames(wide_df)[-1],  # Select all group columns
                   v.names="Value",
                   idvar="Time",
                   timevar="Group",
                   times=1:num_groups,
                   direction="long")

# Ensure no unexpected columns
long_df <- long_df[, c("Time", "Group", "Value")]



# Display first few rows
head(long_df)
```

5. **Evaluate & Compare**

In reality, you might do this comparison inside the partner's environment to confirm your synthetic data is a close match. For demonstration, we'll just compare directly here.

```r
# Step 5: Evaluate

# A) Check Means & SDs
synth_means <- colMeans(df_synth[, c("X1","X2","X3","X4")])
synth_sds   <- apply(df_synth[, c("X1","X2","X3","X4")], 2, sd)

cat("Original (Collected) Means:\n", round(means, 3), "\n")
#> Original (Collected) Means:
#>  -0.652 7.764 2.383 1.312
cat("Synthetic Means:\n", round(synth_means, 3), "\n\n")
#> Synthetic Means:
#>  -0.627 7.882 2.441 1.315

cat("Original (Collected) SDs:\n", round(sds, 3), "\n")
```

```
#> Original (Collected) SDs:
#>  1.448 4.937 1.16 1.102
cat("Synthetic SDs:\n", round(synth_sds, 3), "\n\n")
#> Synthetic SDs:
#>  1.467 4.939 1.159 1.103

# B) Check Correlation
synth_cor <- cor(df_synth[, c("X1","X2","X3","X4")])
cat("Original (Collected) Correlation Matrix:\n")
#> Original (Collected) Correlation Matrix:
print(round(R, 3))
#>        X1     X2     X3     X4
#> X1  1.000 -0.687  0.395 -0.704
#> X2 -0.687  1.000  0.130  0.753
#> X3  0.395  0.130  1.000 -0.366
#> X4 -0.704  0.753 -0.366  1.000
cat("\nSynthetic Correlation Matrix:\n")
#>
#> Synthetic Correlation Matrix:
print(round(synth_cor, 3))
#>        X1     X2     X3     X4
#> X1  1.000 -0.693  0.402 -0.704
#> X2 -0.693  1.000  0.114  0.756
#> X3  0.402  0.114  1.000 -0.376
#> X4 -0.704  0.756 -0.376  1.000
```

You should see that the synthetic dataset's means, SDs, and correlation matrix are **very close** to the manually collected values from `df_original`.

```
# Histograms or density plots
par(mfrow = c(2,2))
hist(df_synth$X1, main="X1 Synthetic", col="lightblue", breaks=30)
hist(df_synth$X2, main="X2 Synthetic", col="lightblue", breaks=30)
hist(df_synth$X3, main="X3 Synthetic", col="lightblue", breaks=30)
hist(df_synth$X4, main="X4 Synthetic", col="lightblue", breaks=30)
```

```r
# Pairwise correlation scatterplots
library(PerformanceAnalytics)
chart.Correlation(df_synth[, c("X1","X2","X3","X4")],
                  histogram=TRUE, pch=19)
```

43.3.3 Synthpop

The easiest way to create synthetic data is to use the `synthpop` package.

```r
library(synthpop)
library(tidyverse)
library(performance)

# library(effectsize)
# library(see)
# library(patchwork)
# library(knitr)
# library(kableExtra)

head(iris)
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#> 1          5.1         3.5          1.4         0.2  setosa
#> 2          4.9         3.0          1.4         0.2  setosa
#> 3          4.7         3.2          1.3         0.2  setosa
#> 4          4.6         3.1          1.5         0.2  setosa
#> 5          5.0         3.6          1.4         0.2  setosa
#> 6          5.4         3.9          1.7         0.4  setosa

synthpop::codebook.syn(iris)
#> $tab
#>       variable   class nmiss perctmiss ndistinct
#> 1 Sepal.Length numeric     0         0        35
#> 2  Sepal.Width numeric     0         0        23
#> 3 Petal.Length numeric     0         0        43
#> 4  Petal.Width numeric     0         0        22
#> 5      Species  factor     0         0         3
#>                              details
#> 1                  Range: 4.3 - 7.9
#> 2                    Range: 2 - 4.4
#> 3                    Range: 1 - 6.9
#> 4                  Range: 0.1 - 2.5
#> 5 'setosa' 'versicolor' 'virginica'
#>
#> $labs
#> NULL

syn_df <- syn(iris, seed = 3)
#>
#> Synthesis
#> -----------
#>  Sepal.Length Sepal.Width Petal.Length Petal.Width Species

# check for replciated uniques
replicated.uniques(syn_df, iris)
```

```
#> $replications
#>   [1]  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE
#>  [13]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
#>  [25] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
#>  [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
#>  [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
#>  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE
#>  [73] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
#>  [85] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE
#>  [97] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE
#> [109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
#> [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
#> [133] FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
#> [145] FALSE FALSE FALSE FALSE FALSE FALSE
#>
#> $no.uniques
#> [1] 148
#>
#> $no.replications
#> [1] 17
#>
#> $per.replications
#> [1] 11.33333


# remove replicated uniques and adds a FAKE_DATA label
# (in case a person can see his or own data in
# the replicated data by chance)

syn_df_sdc <- sdc(syn_df, iris,
                  label = "FAKE_DATA",
                  rm.replicated.uniques = T)
#> no. of replicated uniques: 17
```

```
iris |>
    GGally::ggpairs()
```

```
syn_df$syn |>
    GGally::ggpairs()
```

```
lm_ori <- lm(Sepal.Length ~ Sepal.Width + Petal.Length , data = iris)
# performance::check_model(lm_ori)
summary(lm_ori)
#>
#> Call:
```

```
#> lm(formula = Sepal.Length ~ Sepal.Width + Petal.Length, data = iris)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -0.96159 -0.23489  0.00077  0.21453  0.78557
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   2.24914    0.24797    9.07 7.04e-16 ***
#> Sepal.Width   0.59552    0.06933    8.59 1.16e-14 ***
#> Petal.Length  0.47192    0.01712   27.57  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.3333 on 147 degrees of freedom
#> Multiple R-squared:  0.8402, Adjusted R-squared:  0.838
#> F-statistic: 386.4 on 2 and 147 DF,  p-value: < 2.2e-16

lm_syn <- lm(Sepal.Length ~ Sepal.Width + Petal.Length , data = syn_df$syn)
# performance::check_model(lm_syn)
summary(lm_syn)
#>
#> Call:
#> lm(formula = Sepal.Length ~ Sepal.Width + Petal.Length, data = syn_df$syn)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -0.79165 -0.22790 -0.01448  0.15893  1.13360
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   2.96449    0.24538  12.081  < 2e-16 ***
#> Sepal.Width   0.39214    0.06816   5.754  4.9e-08 ***
#> Petal.Length  0.45267    0.01743  25.974  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.3658 on 147 degrees of freedom
#> Multiple R-squared:  0.8246, Adjusted R-squared:  0.8222
#> F-statistic: 345.6 on 2 and 147 DF,  p-value: < 2.2e-16
```

Open data can be assessed for its utility in two distinct ways:

1. **General Utility**: This refers to the broad resemblances within the dataset, allowing for preliminary data exploration.

2. **Specific Utility**: This focuses on the comparability of models derived from synthetic and original datasets, emphasizing analytical reproducibility.

For General utility

```
compare(syn_df, iris)
```

Specific utility

```
# just like regular lm, but for synthetic data
lm_syn <- lm.synds(Sepal.Length ~ Sepal.Width + Petal.Length , data = syn_df)
compare(lm_syn, iris)
#>
#> Call used to fit models to the data:
#> lm.synds(formula = Sepal.Length ~ Sepal.Width + Petal.Length,
#>     data = syn_df)
#>
#> Differences between results based on synthetic and observed data:
#>              Synthetic  Observed        Diff Std. coef diff CI overlap
#> (Intercept)  2.9644900 2.2491402  0.71534988       2.884829  0.2640608
#> Sepal.Width  0.3921429 0.5955247 -0.20338187      -2.933611  0.2516161
#> Petal.Length 0.4526695 0.4719200 -0.01925058      -1.124602  0.7131064
#>
#> Measures for one synthesis and 3 coefficients
#> Mean confidence interval overlap:  0.4095944
#> Mean absolute std. coef diff:  2.314347
#>
#> Mahalanobis distance ratio for lack-of-fit (target 1.0): 3.08
#> Lack-of-fit test: 9.23442; p-value 0.0263 for test that synthesis model is
#> compatible with a chi-squared test with 3 degrees of freedom.
#>
#> Confidence interval plot:
```

```
# summary(lm_syn)
```

You basically want your lack-of-fit test to be non-significant.

Appendix A

Appendix

A.1 Git

Cheat Sheet

Cheat Sheet in different languages

Learn Git

Interactive Cheat Sheet

Ultimate Guide of Git and GitHub for R user

- Setting up Git: `git config` with `--global` option to configure user name, email, editor, etc.
- Creating a repository: `git init` to initialize a repo. Git stores all of its repo data in the `.git` directory.
- Tracking changes:
  - `git status` shows the status of the repo
    - File are stored in the project's working directory (which users see)
    - The staging area (where the next commit is being built)
    - local repo is where commits are permanently recorded
  - `git add` put files in the staging area
  - `git commit` saves the staged content as a new commit in the local repo.
    - `git commit -m "your own message"` to give a messages for the purpose of your commit.

- History
  - `git diff` shows differences between commits
  - `git checkout` recovers old version of fields
    - `git checkout HEAD` to go to the last commit
    - `git checkout <unique ID of your commit>` to go to such commit
- Ignoring
  - `.gitignore` file tells Git what files to ignore
  - `cat . gitignore *.dat results/` ignore files ending with "dat" and folder "results".
- Remotes in GitHub
  - A local git repo can be connected to one or more remote repos.
  - Use the HTTPS protocol to connect to remote repos
  - `git push` copies changes from a local repo to a remote repo
  - `git pull` copies changes from a remote repo to a local repo
- Collaborating
  - `git clone` copies remote repo to create a local repo with a remote called `origin` automatically set up
- Branching
  - `git check - b <new-branch-name`
  - `git checkout master` to switch to master branch.
- Conflicts
  - occur when 2 or more people change the same lines of the same file
  - the version control system does not allow to overwrite each other's changes blindly, but highlights conflicts so that they can be resolved.
- Licensing
  - People who incorporate General Public License (GPL'd) software into their won software must make their software also open under the GPL license; most other open licenses do not require this.
  - The Creative Commons family of licenses allow people to mix and match requirements and restrictions on attribution, creation of derivative works, further sharing and commercialization.
- Citation:

  - Add a CITATION file to a repo to explain how you want others to cite your work.

- Hosting
  - Rules regarding intellectual property and storage of sensitive info apply no matter where code and data are hosted.

A.2 Short-cut

These are shortcuts that you probably you remember when working with R. Even though it might take a bit of time to learn and use them as your second nature, but they will save you a lot of time.
Just like learning another language, the more you speak and practice it, the more comfortable you are speaking it.

function	short-cut
navigate folders in console	`" " + tab`
pull up short-cut cheat sheet	`ctrl + shift + k`
go to file/function (everything in your project)	`ctrl + .`
search everything	`cmd + shift + f`
navigate between tabs	`Crtl + shift + .`
type function faster	`snip + shift + tab`
type faster	`use tab for fuzzy match`
`cmd + up`	
`ctrl + .`	

Sometimes you can't stage a folder because it's too large. In such case, use `Terminal` pane in Rstudio then type `git add -A` to stage all changes then commit and push like usual.

A.3 Function short-cut

apply one function to your data to create a new variable: `mutate(mod=map(data,function))`
instead of using `i in 1:length(object)`: `for (i in seq_along(object))`
apply multiple function: `map_dbl`
apply multiple function to multiple variables:`map2`
`autoplot(data)` plot times series data
`mod_tidy = linear(reg) %>% set_engine('lm') %>% fit(price ~ ., data=data)` fit lm model. It could also fit other models (stan, spark, glmnet, keras)

- Sometimes, data-masking will not be able to recognize whether you're calling from environment or data variables. To bypass this, we use `.data$variable` or `.env$variable`. For example `data %>% mutate(x=.env$variable/.data$variable`

- Problems with data-masking:

  - Unexpected masking by data-var: Use `.data` and `.env` to disambiguate

  - Data-var cant get through:

  - Tunnel data-var with {{}} + Subset `.data` with [[]]
- Passing Data-variables through arguments

```
library("dplyr")
mean_by <- function(data,by,var){
    data %>%
        group_by({{{by}}}) %>%
        summarise("{{var}}":=mean({{var}})) # new name for each var will be created by
}

mean_by <- function(data,by,var){
    data %>%
        group_by({{{by}}}) %>%
        summarise("{var}":=mean({{var}})) # use single {} to glue the string, but hard
}
```

- Trouble with selection:

```
library("purrr")
name <- c("mass","height")
starwars %>% select(name) # Data-var. Here you are referring to variable named "name"

starwars %>% select(all_of((name))) # use all_of() to disambiguate when

averages <- function(data,vars){ # take character vectors with all_of()
    data %>%
        select(all_of(vars)) %>%
        map_dbl(mean,na.rm=TRUE)
}

x = c("Sepal.Length","Petal.Length")
iris %>% averages(x)
```

```
# Another way
averages <- function(data,vars){ # Tunnel selectiosn with {{}}
    data %>%
        select({{vars}}) %>%
        map_dbl(mean,na.rm=TRUE)
}

x = c("Sepal.Length","Petal.Length")
iris %>% averages(x)
```

A.4 Citation

include a citation by `[@Farjam_2015]`

cite packages used in this session

`package=ls(sessionInfo()$loadedOnly) for (i in package){print(toBibtex(citation(i)))}`

```
package=ls(sessionInfo()$loadedOnly)
for (i in package){
    print(toBibtex(citation(i)))
    }
```

A.5 Install all necessary packages/libaries on your local machine

Get a list of packages you need to install from this book (or your local device)

```
installed <- as.data.frame(installed.packages())

head(installed)
#>         Package                           LibPath Version Priority
#> abind     abind C:/Program Files/R/R-4.2.3/library   1.4-5     <NA>
#> ade4       ade4 C:/Program Files/R/R-4.2.3/library  1.7-22     <NA>
#> admisc   admisc C:/Program Files/R/R-4.2.3/library    0.33     <NA>
#> AER         AER C:/Program Files/R/R-4.2.3/library  1.2-10     <NA>
#> afex       afex C:/Program Files/R/R-4.2.3/library   1.3-0     <NA>
#> agridat agridat C:/Program Files/R/R-4.2.3/library    1.21     <NA>
#>                                                                                                  Depends
#> abind                                                                                       R (>= 1.5.0)
#> ade4                                                                                         R (>= 2.10)
#> admisc                                                                                      R (>= 3.5.0)
#> AER     R (>= 3.0.0), car (>= 2.0-19), lmtest, sandwich (>= 2.4-0),\nsurvival (>= 2.37-5), zoo
#> afex                                                                R (>= 3.5.0), lme4 (>= 1.1-8)
#> agridat                                                                                        <NA>
```

```
#>                                                                                    Imp
#> abind                                                                     methods, u
#> ade4                graphics, grDevices, methods, stats, utils, MASS, pixmap, sp,\n
#> admisc                                                                          meti
#> AER                                                          stats, Formula (>= 0.
#> afex    pbkrtest (>= 0.4-1), lmerTest (>= 3.0-0), car, reshape2,\nstats, methods, u
#> agridat
#>                   LinkingTo
#> abind                  <NA>
#> ade4    Rcpp, RcppArmadillo
#> admisc                 <NA>
#> AER                    <NA>
#> afex                   <NA>
#> agridat                <NA>
#>
#> abind
#> ade4
#> admisc
#> AER
#> afex    emmeans (>= 1.4), coin, xtable, parallel, plyr, optimx,\nnloptr, knitr, rma
#> agridat                     AER, agricolae, betareg, broom, car, coin, corrgram, des
#>         Enhances       License License_is_FOSS License_restricts_use OS_type
#> abind       <NA>   LGPL (>= 2)            <NA>                  <NA>    <NA>
#> ade4        <NA>    GPL (>= 2)            <NA>                  <NA>    <NA>
#> admisc      <NA>    GPL (>= 3)            <NA>                  <NA>    <NA>
#> AER         <NA> GPL-2 | GPL-3            <NA>                  <NA>    <NA>
#> afex        <NA>    GPL (>= 2)            <NA>                  <NA>    <NA>
#> agridat     <NA>  CC BY-SA 4.0            <NA>                  <NA>    <NA>
#>         MD5sum NeedsCompilation Built
#> abind     <NA>               no 4.2.0
#> ade4      <NA>              yes 4.2.3
#> admisc    <NA>              yes 4.2.3
#> AER       <NA>               no 4.2.3
#> afex      <NA>               no 4.2.3
#> agridat   <NA>               no 4.2.3

write.csv(installed, file.path(getwd(),'installed.csv'))
```

After having the `installed.csv` file on your new or local machine, you can just install the list of packages

```
# import the list of packages
installed <- read.csv('installed.csv')

# get the list of packages that you have on your device
baseR <- as.data.frame(installed.packages())
```

```r
# install only those that you don't have
install.packages(setdiff(installed, baseR))
```

Appendix B

Bookdown cheat sheet

```
# to see non-scientific notation a result
format(12e-17, scientific = FALSE)
#> [1] "0.00000000000000012"
```

B.1 Operation

R commands to do derivatives of a defined function Taking derivatives in R involves using the `expression, D,` and `eval` functions. You wrap the function you want to take the derivative of in expression(), then use D, then eval as follows.

simple example

```
#define a function
f=expression(sqrt(x))

#take the first derivative
df.dx=D(f,'x')
df.dx
#> 0.5 * x^-0.5

#take the second derivative
d2f.dx2=D(D(f,'x'),'x')
d2f.dx2
#> 0.5 * (-0.5 * x^-1.5)
```

Evaluate

- The first argument passed to eval is the expression you want to evaluate

- the second is a list containing the values of all quantities that are not defined elsewhere.

```
#evaluate the function at a given x
eval(f,list(x=3))
#> [1] 1.732051

#evaluate the first derivative at a given x
eval(df.dx,list(x=3))
#> [1] 0.2886751

#evaluate the second derivative at a given x
eval(d2f.dx2,list(x=3))
#> [1] -0.04811252
```

B.2 Math Expression/ Syntax

Full list

Aligning equations

```
\begin{aligned}
a & = b \\
X &\sim {Norm}(10, 3) \\
5 & \le 10
\end{aligned}
```

$$\begin{aligned}
a & = b \\
X &\sim {Norm}(10, 3) \\
5 & \le 10
\end{aligned}$$

Cross-reference equation

```
\begin{equation}
a = b
(\#eq:test)
\end{equation}
```

$$a = b \tag{B.1}$$

to refer in a sentence (B.1) (`\@ref(eq:test)`)

Math Syntax	Notation
`$\pm$`	$\pm$

Math Syntax	Notation
`$\ge$`	$\ge$
`$\le$`	$\le$
`$\neq$`	$\neq$
`$\equiv$`	$\equiv$
`$^\circ$`	$^\circ$
`$\times$`	$\times$
`$\cdot$`	$\cdot$
`$\leq$`	$\leq$
`$\geq$`	$\geq$
`\propto`	$\propto$
`$\subset$`	$\subset$
`$\subseteq$`	$\subseteq$
`$\leftarrow$`	$\leftarrow$
`$\rightarrow$`	$\rightarrow$
`$\Leftarrow$`	$\Leftarrow$
`$\Rightarrow$`	$\Rightarrow$
`$\approx$`	$\approx$
`$\mathbb{R}$`	$\mathbb{R}$
`$\sum_{n=1}^{10} n^2$`	$\sum_{n=1}^{10} n^2$
`$$\sum_{n=1}^{10} n^2$$`	$\sum_{n=1}^{10} n^2$
`$x^{n}$`	x^{n}
`$x_{n}$`	x_{n}
`$\overline{x}$`	$\overline{x}$
`$\hat{x}$`	$\hat{x}$
`$\tilde{x}$`	$\tilde{x}$
`\check{}`	$\check{}$
`\underset{\gamma}{\operatorname{argmin}}`	$\underset{\gamma}{\operatorname{argmin}}$
`$\frac{a}{b}$`	$\frac{a}{b}$
`$\frac{a}{b}$`	$\frac{a}{b}$
`$\displaystyle \frac{a}{b}$`	$\displaystyle \frac{a}{b}$
`$\binom{n}{k}$`	$\binom{n}{k}$
`$x_{1} + x_{2} + \cdots + x_{n}$`	$x_{1} + x_{2} + \cdots + x_{n}$
`$x_{1}, x_{2}, \dots, x_{n}$`	$x_{1}, x_{2}, \dots, x_{n}$
`\mathbf{x} = \langle x_{1}, x_{2}, \dots, x_{n}\rangle$`	$\mathbf{x} = \langle x_{1}, x_{2}, \dots, x_{n}\rangle$
`$x \in A$`	$x \in A$
`$\|A\|$`	$\|A\|$
`$x \in A$`	$x \in A$
`$x \subset B$`	$x \subset B$
`$x \subseteq B$`	$x \subseteq B$
`$A \cup B$`	$A \cup B$

Math Syntax	Notation
`$A \cap B$`	$A \cap B$
`$X \sim Binom(n, \pi)$`	$X \sim Binom(n, \pi)$
`$\mathrm{P}(X \le x) = \text{pbinom}(x, n, \pi)$`	$\mathrm{P}(X \le x) = \text{pbinom}(x, n, \pi)$
`$P(A \mid B)$`	$P(A \mid B)$
`$\mathrm{P}(A \mid B)$`	$\mathrm{P}(A \mid B)$
`$\{1, 2, 3\}$`	$\{1, 2, 3\}$
`$\sin(x)$`	$\sin(x)$
`$\log(x)$`	$\log(x)$
`$\int_{a}^{b}$`	$\int_{a}^{b}$
`$\left(\int_{a}^{b} f(x) \; dx\right)$`	$\left(\int_{a}^{b} f(x) \; dx\right)$
`$\left[\int_{\-infty}^{\infty} f(x) \; dx\right]$`	$\left[\int_{-\infty}^{\infty} f(x) \; dx\right]$
`$\left. F(x) \right\|_{a}^{b}$`	$\left. F(x) \right\|_{a}^{b}$
`$\sum_{x = a}^{b} f(x)$`	$\sum_{x = a}^{b} f(x)$
`$\prod_{x = a}^{b} f(x)$`	$\prod_{x = a}^{b} f(x)$
`$\lim_{x \to \infty} f(x)$`	$\lim_{x \to \infty} f(x)$
`$\displaystyle \lim_{x \to \infty} f(x)$`	$\displaystyle \lim_{x \to \infty} f(x)$
Greek Letters	
`$\alpha A$`	αA
`$\beta B$`	βB
`$\gamma \Gamma$`	$\gamma \Gamma$
`$\delta \Delta$`	$\delta \Delta$
`$\epsilon \varepsilon E$`	$\epsilon \varepsilon E$
`$\zeta Z \sigma $`	$\zeta Z \sigma$
`$\eta H$`	ηH
`$\theta \vartheta \Theta$`	$\theta \vartheta \Theta$
`$\iota I$`	ιI
`$\kappa K$`	κK
`$\lambda \Lambda$`	$\lambda \Lambda$
`$\mu M$`	μM
`$\nu N$`	νN
`$\xi\Xi$`	$\xi \Xi$
`$o O$`	$o O$
`$\pi \Pi$`	$\pi \Pi$
`$\rho\varrho P$`	$\rho \varrho P$
`$\sigma \Sigma$`	$\sigma \Sigma$
`$\tau T$`	τT
`$\upsilon \Upsilon$`	$\upsilon \Upsilon$
`$\phi \varphi \Phi$`	$\phi \varphi \Phi$
`$\chi X$`	χX

Math Syntax	Notation
`$\psi \Psi$`	$\psi\Psi$
`$\omega \Omega$`	$\omega\Omega$
`$\cdot$`	$\cdot$
`$\cdots$`	$\cdots$
`$\ddots$`	$\ddots$
`$\ldots$`	$\ldots$

Limit `P(\lim_{n\to \infty}\bar{X}_n =\mu) =1`

$$P(\lim_{n\to\infty} \bar{X}_n = \mu) = 1$$

Matrices

```
$$\begin{array}
{rrr}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{array}
$$
```

$$\begin{array}{rrr}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{array}$$

```
$$\mathbf{X} = \left[\begin{array}
{rrr}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{array}\right]
$$
```

$$\mathbf{X} = \left[\begin{array}{rrr}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{array}\right]$$

Aligning Equations

Aligning Equations with Comments

```
\begin{aligned}
    3+x &=4 && \text{(Solve for} x \text{.)}\\
    x &=4-3 && \text{(Subtract 3 from both sides.)}\\
    x &=1   && \text{(Yielding the solution.)}
\end{aligned}
```

$$\begin{aligned}
    3+x &=4 && \text{(Solve for} x \text{.)}\\
    x &=4-3 && \text{(Subtract 3 from both sides.)}\\
    x &=1   && \text{(Yielding the solution.)}
\end{aligned}$$

B.2.1 Statistics Notation

```
$$
f(y|N,p) = \frac{N!}{y!(N-y)!}\cdot p^y \cdot (1-p)^{N-y} = {{N}\choose{y}} \cdot p^y \
$$
```

$$f(y|N,p) = \frac{N!}{y!(N-y)!}\cdot p^y \cdot (1-p)^{N-y} = {{N}\choose{y}} \cdot p^y \cdot (1-p)^{N-y}$$

```
\begin{cases}
\frac{1}{b-a}&\text{for $x\in[a,b]$}\\
0&\text{otherwise}\\
\end{cases}
```

$$\begin{cases}
\frac{1}{b-a}&\text{for $x\in[a,b]$}\\
0&\text{otherwise}\\
\end{cases}$$

B.3 Table

```
+---------------+---------------+--------------------+
| Fruit         | Price         | Advantages         |
+===============+===============+====================+
| *Bananas*     | $1.34         | - built-in wrapper |
|               |               | - bright color     |
+---------------+---------------+--------------------+
| Oranges       | $2.10         | - cures scurvy     |
|               |               | - **tasty**        |
+---------------+---------------+--------------------+
```

Fruit	Price	Advantages
Bananas	$1.34	• built-in wrapper • bright color
Oranges	$2.10	• cures scurvy • **tasty**

```
(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y}
```

$(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y}$

Bibliography

(2020).

Aaronson, D., Barrow, L., and Sander, W. (2007). Teachers and student achievement in the chicago public high schools. *Journal of labor Economics*, 25(1):95–135.

Abadie, A. (2021). Using synthetic controls: Feasibility, data requirements, and methodological aspects. *Journal of Economic Literature*, 59(2):391–425.

Abadie, A., Angrist, J., and Imbens, G. (2002). Instrumental variables estimates of the effect of subsidized training on the quantiles of trainee earnings. *Econometrica*, 70(1):91–117.

Abadie, A., Athey, S., Imbens, G. W., and Wooldridge, J. M. (2023). When should you adjust standard errors for clustering? *The Quarterly Journal of Economics*, 138(1):1–35.

Abadie, A., Diamond, A., and Hainmueller, J. (2010). Synthetic control methods for comparative case studies: Estimating the effect of california's tobacco control program. *Journal of the American statistical Association*, 105(490):493–505.

Abadie, A. and Imbens, G. W. (2016). Matching on the estimated propensity score. *Econometrica*, 84(2):781–807.

Abadie, A. and L'hour, J. (2021). A penalized synthetic control estimator for disaggregated data. *Journal of the American Statistical Association*, 116(536):1817–1834.

Acemoglu, D., Naidu, S., Restrepo, P., and Robinson, J. A. (2019). Democracy does cause growth. *Journal of political economy*, 127(1):47–100.

Acharya, S. (1993). Value of latent information: alternative event study methods. *The Journal of Finance*, 48(1):363–385.

Adalja, A., Liaukonytė, J., Wang, E., and Zhu, X. (2023). Gmo and non-gmo labeling effects: Evidence from a quasi-natural experiment. *Marketing Science*, 42(2):233–250.

Agarwal, A., Dahleh, M., Shah, D., and Shen, D. (2023). Causal matrix completion. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 3821–3826. PMLR.

Agrawal, J. and Kamakura, W. A. (1995). The economic worth of celebrity endorsers: An event study analysis. *Journal of marketing*, 59(3):56–62.

Ahrens, H. and Pincus, R. (1981). On two measures of unbalancedness in a one-way model and their relation to efficiency. *Biometrical Journal*, 23(3):227–235.

Aiken, L. S. and West, S. G. (2005). Interaction effects. *Encyclopedia of statistics in behavioral science*.

Akca, S. and Rao, A. (2020). Value of aggregators. *Marketing Science*, 39(5):893–922.

Allison, P. D. and Waterman, R. P. (2002). 7. fixed-effects negative binomial regression models. *Sociological methodology*, 32(1):247–265.

Altonji, J. G., Elder, T. E., and Taber, C. R. (2005). Selection on observed and unobserved variables: Assessing the effectiveness of catholic schools. *Journal of political economy*, 113(1):151–184.

Amemiya, T. (1971). The estimation of the variances in a variance-components model. *International Economic Review*, pages 1–13.

Amemiya, T. (1984). Tobit models: A survey. *Journal of econometrics*, 24(1-2):3–61.

Amemiya, T. and MaCurdy, T. E. (1986). Instrumental-variable estimation of an error-components model. *Econometrica: Journal of the Econometric Society*, pages 869–880.

Anderson, J. E., Larch, M., and Yotov, Y. V. (2015). Growth and trade with frictions: A structural estimation framework. Technical report, National Bureau of Economic Research.

Anderson, M. L. (2014). Subways, strikes, and slowdowns: The impacts of public transit on traffic congestion. *American Economic Review*, 104(9):2763–2796.

Anderson, T. W. and Darling, D. A. (1952). Asymptotic theory of certain" goodness of fit" criteria based on stochastic processes. *The annals of mathematical statistics*, pages 193–212.

Anderson, T. W. and Rubin, H. (1949). Estimation of the parameters of a single equation in a complete system of stochastic equations. *The Annals of mathematical statistics*, 20(1):46–63.

Andrews, D. F. (1974). A robust method for multiple linear regression. *Technometrics*, 16(4):523–531.

Andrews, D. W. and Marmer, V. (2008). Exactly distribution-free inference in instrumental variables regression with possibly weak instruments. *Journal of Econometrics*, 142(1):183–200.

Andrews, D. W., Moreira, M. J., and Stock, J. H. (2006). Optimal two-sided invariant similar tests for instrumental variables regression. *Econometrica*, 74(3):715–752.

Andrews, D. W., Moreira, M. J., and Stock, J. H. (2008). Efficient two-sided nonsimilar invariant tests in iv regression with weak instruments. *Journal of Econometrics*, 146(2):241–254.

Andrews, I., Stock, J. H., and Sun, L. (2019). Weak instruments in instrumental variables regression: Theory and practice. *Annual Review of Economics*, 11:727–753.

Angrist, J. and Imbens, G. (1991). Sources of identifying information in evaluation models.

Angrist, J. and Kolesár, M. (2023). One instrument to rule them all: The bias and coverage of just-id iv. *Journal of Econometrics*.

Angrist, J. and Kolesár, M. (2024). One instrument to rule them all: The bias and coverage of just-id iv. *Journal of Econometrics*, 240(2):105398.

Angrist, J. D., Graddy, K., and Imbens, G. W. (2000). The interpretation of instrumental variables estimators in simultaneous equations models with an application to the demand for fish. *The Review of Economic Studies*, 67(3):499–527.

Angrist, J. D. and Imbens, G. W. (1995). Two-stage least squares estimation of average causal effects in models with variable treatment intensity. *Journal of the American statistical Association*, 90(430):431–442.

Angrist, J. D. and Krueger, A. B. (2001). Instrumental variables and the search for identification: From supply and demand to natural experiments. *Journal of Economic perspectives*, 15(4):69–85.

Angrist, J. D. and Lavy, V. (1999). Using maimonides' rule to estimate the effect of class size on scholastic achievement. *The Quarterly journal of economics*, 114(2):533–575.

Angrist, J. D. and Pischke, J.-S. (2009). *Mostly harmless econometrics: An empiricist's companion*. Princeton university press.

Arellano, M. (1987). Computing robust standard errors for within-groups estimators. *Oxford bulletin of Economics and Statistics*, 49(4):431–434.

Arkhangelsky, D., Athey, S., Hirshberg, D. A., Imbens, G. W., and Wager, S. (2019). Synthetic difference in differences. Technical report, National Bureau of Economic Research.

Arkhangelsky, D., Athey, S., Hirshberg, D. A., Imbens, G. W., and Wager, S. (2021). Synthetic difference-in-differences. *American Economic Review*, 111(12):4088–4118.

Arkhangelsky, D. and Hirshberg, D. (2023). Large-sample properties of the synthetic control method under selection on unobservables. *arXiv preprint arXiv:2311.13575*, 2.

Aronow, P. M., Baron, J., and Pinson, L. (2019). A note on dropping experimental subjects who fail a manipulation check. *Political Analysis*, 27(4):572–589.

Ashenfelter, O. and Card, D. (1985). Using the longitudinal structure of earnings to estimate the effect of training programs. *The Review of Economics and Statistics*, 67(4):648.

Assmann, S. F., Hosmer, D. W., Lemeshow, S., and Mundt, K. A. (1996). Confidence intervals for measures of interaction. *Epidemiology*, pages 286–290.

Athey, S., Bayati, M., Doudchenko, N., Imbens, G., and Khosravi, K. (2021). Matrix completion methods for causal panel data models. *Journal of the American Statistical Association*, 116(536):1716–1730.

Athey, S. and Imbens, G. W. (2006). Identification and inference in nonlinear difference-in-differences models. *Econometrica*, 74(2):431–497.

Athey, S. and Imbens, G. W. (2019). Machine learning methods that economists should know about. *Annual Review of Economics*, 11:685–725.

Athey, S. and Imbens, G. W. (2022). Design-based analysis in difference-in-differences settings with staggered adoption. *Journal of Econometrics*, 226(1):62–79.

Auffhammer, M. and Kellogg, R. (2011). Clearing the air? the effects of gasoline content regulation on air quality. *American Economic Review*, 101(6):2687–2722.

Austin, P. C. (2011). Optimal caliper widths for propensity-score matching when estimating differences in means and differences in proportions in observational studies. *Pharmaceutical statistics*, 10(2):150–161.

Azoulay, P., Fons-Rosen, C., and Zivin, J. S. G. (2019). Does science advance one funeral at a time? *American Economic Review*, 109(8):2889–2920.

Babcock, P. (2010). Real costs of nominal grade inflation? new evidence from student course evaluations. *Economic inquiry*, 48(4):983–996.

Bai, J. and Ng, S. (2002). Determining the number of factors in approximate factor models. *Econometrica*, 70(1):191–221.

Baicker, K. and Svoronos, T. (2019). Testing the validity of the single interrupted time series design. Technical report, National Bureau of Economic Research.

Bajari, P., Hong, H., Park, M., and Town, R. (2011). Regression discontinuity designs with an endogenous forcing variable and an application to contracting in health care. Technical report, National Bureau of Economic Research.

Baker, A. C., Larcker, D. F., and Wang, C. C. (2022). How much should we trust staggered difference-in-differences estimates? *Journal of Financial Economics*, 144(2):370–395.

Balasubramanian, S. K., Mathur, I., and Thakur, R. (2005). The impact of high-quality firm achievements on shareholder value: focus on malcolm baldrige and jd power and associates awards. *Journal of the Academy of Marketing Science*, 33(4):413–422.

Balestra, P. and Varadharajan-Krishnakumar, J. (1987). Full information estimations of a system of simultaneous equations with error component structure. *Econometric Theory*, 3(2):223–246.

Baltagi, B. H. (1981). Simultaneous equations with error components. *Journal of econometrics*, 17(2):189–200.

Baltagi, B. H. and Li, Q. (1991). A joint test for serial correlation and random individual effects. *Statistics & Probability Letters*, 11(3):277–280.

Baltagi, B. H. and Li, Q. (1995). Testing ar (1) against ma (1) disturbances in an error component model. *Journal of econometrics*, 68(1):133–151.

Ban, K. and Kedagni, D. (2022). Generalized difference-in-differences models: Robust bounds. *arXiv preprint arXiv:2211.06710*.

Bapna, R., Ramaprasad, J., and Umyarov, A. (2018). Monetizing freemium communities. *Mis Quarterly*, 42(3):719–A4.

Barber, B. M. and Lyon, J. D. (1997). Firm size, book-to-market ratio, and security returns: A holdout sample of financial firms. *The Journal of Finance*, 52(2):875–883.

Bareinboim, E. and Pearl, J. (2014). Transportability from multiple environments with limited experiments: Completeness results. *Advances in neural information processing systems*, 27.

Baron, R. M. and Kenny, D. A. (1986). The moderator–mediator variable distinction in social psychological research: Conceptual, strategic, and statistical considerations. *Journal of personality and social psychology*, 51(6):1173.

Barreca, A. I., Guldi, M., Lindo, J. M., and Waddell, G. R. (2011). Saving babies? revisiting the effect of very low birth weight classification. *The Quarterly Journal of Economics*, 126(4):2117–2123.

Bartlett, M. S. (1947). The use of transformations. *Biometrics*, 3(1):39–52.

Bates, D., Mächler, M., Bolker, B., and Walker, S. (2014). Fitting linear mixed-effects models using lme4.

Bates, D. M. and Watts, D. G. (1980). Relative curvature measures of nonlinearity. *Journal of the Royal Statistical Society: Series B (Methodological)*, 42(1):1–16.

Bates, D. M. and Watts, D. G. (1981). A relative off set orthogonality convergence criterion for nonlinear least squares. *Technometrics*, 23(2):179–183.

Bauer, D. J. and Curran, P. J. (2005). Probing interactions in fixed and multilevel regression: Inferential and graphical techniques. *Multivariate behavioral research*, 40(3):373–400.

Bauer, D. J., Preacher, K. J., and Gil, K. M. (2006). Conceptualizing and testing random indirect effects and moderated mediation in multilevel models: new procedures and recommendations. *Psychological methods*, 11(2):142.

Baum, C. and Schaffer, M. (2021). Ivreg2h: Stata module to perform instrumental variables estimation using heteroskedasticity-based instruments.

Beaton, A. E. and Tukey, J. W. (1974). The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data. *Technometrics*, 16(2):147–185.

Beerli, A., Ruffner, J., Siegenthaler, M., and Peri, G. (2021). The abolition of immigration restrictions and the performance of firms and workers: evidence from switzerland. *American Economic Review*, 111(3):976–1012.

Bellemare, M. F., Masaki, T., and Pepinsky, T. B. (2017). Lagged explanatory variables and the estimation of causal effect. *The Journal of Politics*, 79(3):949–963.

Belloni, A., Chernozhukov, V., and Hansen, C. (2014). High-dimensional methods and inference on structural and treatment effects. *Journal of Economic Perspectives*, 28(2):29–50.

Ben-Michael, E., Feller, A., and Rothstein, J. (2020). Varying impacts of letters of recommendation on college admissions: Approximate balancing weights for subgroup effects in observational studies. *arXiv preprint arXiv:2008.04394*.

Ben-Michael, E., Feller, A., and Rothstein, J. (2021). The augmented synthetic control method. *Journal of the American Statistical Association*, 116(536):1789–1803.

Ben-Michael, E., Feller, A., and Rothstein, J. (2022). Synthetic controls with staggered adoption. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(2):351–381.

Bendel, R. B. and Afifi, A. A. (1977). Comparison of stopping rules in forward "stepwise" regression. *Journal of the American Statistical association*, 72(357):46–53.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate:

a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300.

Benjamini, Y. and Hochberg, Y. (2000). On the adaptive control of the false discovery rate in multiple testing with independent statistics. *Journal of educational and Behavioral Statistics*, 25(1):60–83.

Benjamini, Y., Krieger, A. M., and Yekutieli, D. (2006). Adaptive linear step-up procedures that control the false discovery rate. *Biometrika*, 93(3):491–507.

Benjamini, Y. and Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *Annals of statistics*, pages 1165–1188.

Benston, G. J. (1985). The validity of profits-structure studies with particular reference to the ftc's line of business data. *The American Economic Review*, 75(1):37–67.

Bento, A., Kaffine, D., Roth, K., and Zaragoza-Watkins, M. (2014). The effects of regulation in the presence of multiple unpriced externalities: Evidence from the transportation sector. *American Economic Journal: Economic Policy*, 6(3):1–29.

Bera, A. K. and Jarque, C. M. (1981). Efficient tests for normality, homoscedasticity and serial independence of regression residuals: Monte carlo evidence. *Economics letters*, 7(4):313–318.

Bera, A. K., Sosa-Escudero, W., and Yoon, M. (2001). Tests for the error component model in the presence of local misspecification. *Journal of Econometrics*, 101(1):1–23.

Berger, P. G., Ofek, E., and Yermack, D. L. (1997). Managerial entrenchment and capital structure decisions. *The journal of finance*, 52(4):1411–1438.

Berkouwer, S. B. and Dean, J. T. (2022). Credit, attention, and externalities in the adoption of energy efficient technologies by low-income households. *American Economic Review*, 112(10):3291–3330.

Berman, R. and Israeli, A. (2022). The value of descriptive analytics: Evidence from online retailers. *Marketing Science*, 41(6):1074–1096.

Bernard, V. L. (1987). Cross-sectional dependence and problems in inference in market-based accounting research. *Journal of Accounting Research*, pages 1–48.

Bertanha, M., McCallum, A. H., and Seegert, N. (2021). Better bunching, nicer notching. *arXiv preprint arXiv:2101.01170*.

Bertrand, M., Duflo, E., and Mullainathan, S. (2004). How much should we trust differences-in-differences estimates? *The Quarterly journal of economics*, 119(1):249–275.

Bertrand, M. and Mullainathan, S. (2004). Are emily and greg more employable than lakisha and jamal? a field experiment on labor market discrimination. *American economic review*, 94(4):991–1013.

Bhaduri, S. N. (2002). Determinants of corporate borrowing: some evidence from the indian corporate structure. *Journal of Economics and Finance*, 26(2):200–215.

Bhagat, S., Bizjak, J., and Coles, J. L. (1998). The shareholder wealth implications of corporate lawsuits. *Financial Management*, pages 5–27.

Bhattacharya, J. and Vogt, W. B. (2007). Do instrumental variables belong in propensity scores?

Bickel, P. J. and Doksum, K. A. (1981). An analysis of transformations revisited. *Journal of the american statistical association*, 76(374):296–311.

Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.

Blau, D. M. (1999). The effect of income on child development. *Review of Economics and Statistics*, 81(2):261–276.

Blomquist, S., Newey, W. K., Kumar, A., and Liang, C.-Y. (2021). On bunching and identification of the taxable income elasticity. *Journal of Political Economy*, 129(8):2320–2343.

Bloom, H. S. (1984). Accounting for no-shows in experimental evaluation designs. *Evaluation review*, 8(2):225–246.

Bloom, H. S., Orr, L. L., Bell, S. H., Cave, G., Doolittle, F., Lin, W., and Bos, J. M. (1997). The benefits and costs of jtpa title ii-a programs: Key findings from the national job training partnership act study. *Journal of human resources*, pages 549–576.

Blundell, R. and Bond, S. (1998). Initial conditions and moment restrictions in dynamic panel data models. *Journal of econometrics*, 87(1):115–143.

Böckerman, P., Kanninen, O., and Suoniemi, I. (2018). A kink that makes you sick: The effect of sick pay on absence. *Journal of Applied Econometrics*, 33(4):568–579.

Bollen, K. A. and Stine, R. (1990). Direct and indirect effects: Classical and bootstrap estimates of variability. *Sociological methodology*, pages 115–140.

Borah, A. and Tellis, G. J. (2014). Make, buy, or ally? choice of and payoff from announcements of alternate strategies for innovations. *Marketing Science*, 33(1):114–133.

Borusyak, K., Jaravel, X., and Spiess, J. (2021). Revisiting event study designs: Robust and efficient estimation. *arXiv preprint arXiv:2108.12419*.

Bosch, N., Dekker, V., and Strohmaier, K. (2020). A data-driven procedure to determine the bunching window: an application to the netherlands. *International Tax and Public Finance*, 27:951–979.

Bottomley, C., Scott, J. A. G., and Isham, V. (2019). Analysing interrupted time series with a control. *Epidemiologic Methods*, 8(1):20180010.

Bound, J., Brown, C., and Mathiowetz, N. (2001). Measurement error in survey data. In *Handbook of econometrics*, volume 5, pages 3705–3843. Elsevier.

Bound, J., Brown, C. C., Duncan, G., and Rodgers, W. L. (1989). Measurement error in cross-sectional and longitudinal labor market surveys: Results from two validation studies.

Bound, J., Jaeger, D. A., and Baker, R. M. (1995). Problems with instrumental variables estimation when the correlation between the instruments and the endogenous explanatory variable is weak. *Journal of the American statistical association*, 90(430):443–450.

Bowblis, J. R. and Smith, A. C. (2021). Occupational licensing of social services and nursing home quality: A regression discontinuity approach. *ILR Review*, 74(1):199–223.

Box, G. E. and Cox, D. R. (1981). *An analysis of transformations revisited, rebutted*. University of Wisconsin-Madison. Mathematics Research Center.

Box, G. E. and Pierce, D. A. (1970). Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American statistical Association*, 65(332):1509–1526.

Boyd, D. E., Chandy, R. K., and Cunha Jr, M. (2010). When do chief marketing officers affect firm value? a customer power explanation. *Journal of Marketing Research*, 47(6):1162–1176.

Boyd, D. E. and Spekman, R. E. (2008). The market value impact of indirect ties within technology alliances. *Journal of the Academy of Marketing Science*, 36:488–500.

Braun, M. and Schwartz, E. M. (2025). Where a/b testing goes wrong: How divergent delivery affects what online experiments cannot (and can) tell you about how customers respond to advertising. *Journal of Marketing*, 0(0).

Brav, A. and Gompers, P. A. (1997). Myth or reality? the long-run underperformance of initial public offerings: Evidence from venture and nonventure capital-backed companies. *The journal of finance*, 52(5):1791–1821.

Breiman, L. (2001). Random forests. *Machine learning*, 45:5–32.

Breiman, L. (2017). *Classification and regression trees*. Routledge.

Breusch, T. S. (1978). Testing for autocorrelation in dynamic linear models. *Australian economic papers*, 17(31):334–355.

Breusch, T. S., Mizon, G. E., and Schmidt, P. (1989). Efficient estimation using panel data. *Econometrica: Journal of the Econometric Society*, pages 695–700.

Breusch, T. S. and Pagan, A. R. (1979). A simple test for heteroscedasticity and random coefficient variation. *Econometrica: Journal of the econometric society*, pages 1287–1294.

Breusch, T. S. and Pagan, A. R. (1980). The lagrange multiplier test and its applications to model specification in econometrics. *The review of economic studies*, 47(1):239–253.

Brodeur, A., Clark, A. E., Fleche, S., and Powdthavee, N. (2021). Covid-19, lockdowns and well-being: Evidence from google trends. *Journal of public economics*, 193:104346.

Brodeur, A., Valenta, D., Marcoci, A., Aparicio, J. P., Mikola, D., Barbarioli, B., Alexander, R., Deer, L., Stafford, T., Vilhuber, L., et al. (2025). Comparing human-only, ai-assisted, and ai-led teams on assessing research reproducibility in quantitative social science. Technical report, I4R Discussion Paper Series.

Bronnenberg, B. J., Dubé, J.-P., and Sanders, R. E. (2020). Consumer misinformation and the brand premium: A private label blind taste test. *Marketing Science*, 39(2):382–406.

Brown, N., Butts, K., and Westerlund, J. (2023). Simple Difference-in-Differences Estimation in Fixed-T Panels. Papers 2301.11358, arXiv.org.

Brown, S. J. and Warner, J. B. (1980). Measuring security price performance. *Journal of financial economics*, 8(3):205–258.

Brown, S. J. and Warner, J. B. (1985). Using daily stock returns: The case of event studies. *Journal of financial economics*, 14(1):3–31.

Burger, N. E., Kaffine, D. T., and Yu, B. (2014). Did california's hand-held cell phone ban reduce accidents? *Transportation research part A: policy and practice*, 66:162–172.

Burtch, G., Carnahan, S., and Greenwood, B. N. (2018). Can you gig it? an empirical examination of the gig economy and entrepreneurial activity. *Management Science*, 64(12):5497–5520.

Busse, M., Silva-Risso, J., and Zettelmeyer, F. (2006). $1,000 cash back: The pass-through of auto manufacturer promotions. *American Economic Review*, 96(4):1253–1270.

Busse, M. R., Lacetera, N., Pope, D. G., Silva-Risso, J., and Sydnor, J. R. (2013). Estimating the effect of salience in wholesale and retail car markets. *American Economic Review*, 103(3):575–579.

Busse, M. R., Simester, D. I., and Zettelmeyer, F. (2010). "the best price

you'll ever get": The 2005 employee discount pricing promotions in the us automobile industry. *Marketing science*, 29(2):268–290.

Butcher, K. F., McEwan, P. J., and Weerapana, A. (2014). The effects of an anti-grade inflation policy at wellesley college. *Journal of Economic Perspectives*, 28(3):189–204.

Cabral, M., Cui, C., and Dworsky, M. (2022). The demand for insurance and rationale for a mandate: Evidence from workers' compensation insurance. *American economic review*, 112(5):1621–1668.

Cai, Y., Canay, I. A., Kim, D., and Shaikh, A. M. (2022). On the implementation of approximate randomization tests in linear models with a small number of clusters. *Journal of Econometric Methods*, 12(1):85–103.

Callaway, B. and Li, T. (2019). Quantile treatment effects in difference in differences models with panel data. *Quantitative Economics*, 10(4):1579–1618.

Callaway, B. and Sant'Anna, P. H. (2021). Difference-in-differences with multiple time periods. *Journal of econometrics*, 225(2):200–230.

Calonico, S., Cattaneo, M. D., and Farrell, M. H. (2020). Optimal bandwidth choice for robust bias-corrected inference in regression discontinuity designs. *The Econometrics Journal*, 23(2):192–210.

Calonico, S., Cattaneo, M. D., and Titiunik, R. (2014). Robust nonparametric confidence intervals for regression-discontinuity designs. *Econometrica*, 82(6):2295–2326.

Cameron, A. C., Gelbach, J. B., and Miller, D. L. (2008). Bootstrap-based improvements for inference with clustered errors. *The review of economics and statistics*, 90(3):414–427.

Campbell, J. Y., Lo, A. W., MacKinlay, A. C., and Whitelaw, R. F. (1998). The econometrics of financial markets. *Macroeconomic Dynamics*, 2(4):559–562.

Cao, Z. and Sorescu, A. (2013). Wedded bliss or tainted love? stock market reactions to the introduction of cobranded products. *Marketing Science*, 32(6):939–959.

Card, D. and Krueger, A. B. (1993). Minimum wages and employment: A case study of the fast food industry in new jersey and pennsylvania.

Card, D., Lee, D. S., Pei, Z., and Weber, A. (2015). Inference on causal effects in a generalized regression kink design. *Econometrica*, 83(6):2453–2483.

Carhart, M. M. (1997). On persistence in mutual fund performance. *The Journal of finance*, 52(1):57–82.

Carpenter, C. and Dobkin, C. (2009). The effect of alcohol consumption on mortality: regression discontinuity evidence from the minimum drinking age. *American Economic Journal: Applied Economics*, 1(1):164–182.

Carranza, E., Garlick, R., Orkin, K., and Rankin, N. (2022). Job search and hiring with limited information about workseekers' skills. *American Economic Review*, 112(11):3547–3583.

Carrasco, M. and Doukali, M. (2022). Testing overidentifying restrictions with many instruments and heteroscedasticity using regularised jackknife iv. *The Econometrics Journal*, 25(1):71–97.

Cattaneo, M. D., Idrobo, N., and Titiunik, R. (2019). *A practical introduction to regression discontinuity designs: Foundations*. Cambridge University Press.

Cengiz, D., Dube, A., Lindner, A., and Zipperer, B. (2019). The effect of minimum wages on low-wage jobs. *The Quarterly Journal of Economics*, 134(3):1405–1454.

Chabé-Ferret, S. (2015). Analysis of the bias of matching and difference-in-difference under alternative earnings and selection processes. *Journal of Econometrics*, 185(1):110–123.

Chae, I., Ha, J., and Schweidel, D. A. (2023). Paywall suspensions and digital news subscriptions. *Marketing Science*, 42(4):729–745.

Chaney, P. K., Devinney, T. M., and Winer, R. S. (1991). The impact of new product introductions on the market value of firms. *Journal of business*, pages 573–610.

Chaplin, D. D., Cook, T. D., Zurovac, J., Coopersmith, J. S., Finucane, M. M., Vollmer, L. N., and Morris, R. E. (2018). The internal and external validity of the regression discontinuity design: A meta-analysis of 15 within-study comparisons. *Journal of Policy Analysis and Management*, 37(2):403–429.

Chatterjee, S. (2015). Matrix estimation by universal singular value thresholding.

Chen, H., Li, Q., Kaufman, J. S., Wang, J., Copes, R., Su, Y., and Benmarhnia, T. (2018). Effect of air quality alerts on human health: a regression discontinuity analysis in toronto, canada. *The Lancet Planetary Health*, 2(1):e19–e26.

Chen, J. and Roth, J. (2023). Logs with zeros? some problems and solutions. *The Quarterly Journal of Economics*, page qjad054.

Chen, J. and Roth, J. (2024). Logs with zeros? some problems and solutions. *The Quarterly Journal of Economics*, 139(2):891–936.

Chen, X., John, G., Hays, J. M., Hill, A. V., and Geurs, S. E. (2009a). Learning from a service guarantee quasi experiment. *Journal of marketing research*, 46(5):584–596.

Chen, Y., Ganesan, S., and Liu, Y. (2009b). Does a firm's product-recall strategy affect its financial value? an examination of strategic alternatives during product-harm crises. *Journal of Marketing*, 73(6):214–226.

Chen, Y., Liu, Y., and Zhang, J. (2012). When do third-party product reviews affect firm value and what can firms do? the case of media critics and professional movie reviews. *Journal of Marketing*, 76(2):116–134.

Chernozhukov, V., Chetverikov, D., Demirer, M., Duflo, E., Hansen, C., Newey, W., and Robins, J. (2018). Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68.

Chernozhukov, V. and Hansen, C. (2005). An iv model of quantile treatment effects. *Econometrica*, 73(1):245–261.

Chetty, R., Friedman, J. N., and Rockoff, J. E. (2014). Measuring the impacts of teachers i: Evaluating bias in teacher value-added estimates. *American economic review*, 104(9):2593–2632.

Chetty, R., Hendren, N., and Katz, L. F. (2016). The effects of exposure to better neighborhoods on children: New evidence from the moving to opportunity experiment. *American Economic Review*, 106(4):855–902.

Chung, D. J., Steenburgh, T., and Sudhir, K. (2014). Do bonuses enhance sales productivity? a dynamic structural analysis of bonus-based compensation plans. *Marketing Science*, 33(2):165–187.

Cinelli, C., Forney, A., and Pearl, J. (2022). A crash course in good and bad controls. *Sociological Methods & Research*, page 00491241221099552.

Cinelli, C. and Hazlett, C. (2020). Making sense of sensitivity: Extending omitted variable bias. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 82(1):39–67.

Cinelli, C., Kumor, D., Chen, B., Pearl, J., and Bareinboim, E. (2019). Sensitivity analysis of linear structural causal models. In *International conference on machine learning*, pages 1252–1261. PMLR.

Conley, T. G. (1999). Gmm estimation with cross sectional dependence. *Journal of econometrics*, 92(1):1–45.

Conley, T. G. and Taber, C. R. (2011). Inference with "difference in differences" with a small number of policy changes. *The Review of Economics and Statistics*, 93(1):113–125.

Cornwell, T. B., Pruitt, S. W., and Clark, J. M. (2005). The relationship between major-league sports' official sponsorship announcements and the stock prices of sponsoring firms. *Journal of the Academy of Marketing Science*, 33(4):401–412.

Cragg, J. G. and Donald, S. G. (1993). Testing identifiability and specification in instrumental variable models. *Econometric theory*, 9(2):222–240.

Cramér, H. (1999). *Mathematical methods of statistics*, volume 43. Princeton university press.

Cribari-Neto, F. (2004). Asymptotic inference under heteroskedasticity of unknown form. *Computational Statistics & Data Analysis*, 45(2):215–233.

Cribari-Neto, F. and da Silva, W. B. (2011). A new heteroskedasticity-consistent covariance matrix estimator for the linear regression model. *AStA Advances in Statistical Analysis*, 95:129–146.

Cribari-Neto, F., Souza, T. C., and Vasconcellos, K. L. (2007). Inference under heteroskedasticity and leveraged data. *Communications in Statistics—Theory and Methods*, 36(10):1877–1888.

Cullen, J. B., Jacob, B. A., and Levitt, S. D. (2005). The impact of school choice on student outcomes: an analysis of the chicago public schools. *Journal of Public Economics*, 89(5-6):729–760.

Datta, H., Knox, G., and Bronnenberg, B. J. (2018). Changing their tune: How consumers' adoption of online streaming affects music consumption and discovery. *Marketing Science*, 37(1):5–21.

Davidson, R. and MacKinnon, J. G. (1981). Several tests for model specification in the presence of alternative hypotheses. *Econometrica: Journal of the Econometric Society*, pages 781–793.

Davis, L. W. (2008). The effect of driving restrictions on air quality in mexico city. *Journal of Political Economy*, 116(1):38–81.

Davis, L. W. and Kahn, M. E. (2010). International trade in used vehicles: The environmental consequences of nafta. *American Economic Journal: Economic Policy*, 2(4):58–82.

Daw, J. R. and Hatfield, L. A. (2018). Matching and regression to the mean in difference-in-differences analysis. *Health services research*, 53(6):4138–4156.

Dawson, J. F. and Richter, A. W. (2006). Probing three-way interactions in moderated multiple regression: development and application of a slope difference test. *Journal of applied psychology*, 91(4):917.

De Chaisemartin, C. and d'Haultfoeuille, X. (2020). Two-way fixed effects estimators with heterogeneous treatment effects. *American Economic Review*, 110(9):2964–2996.

De Chaisemartin, C. and D'haultfœuille, X. (2023). Two-way fixed effects and differences-in-differences estimators with several treatments. *Journal of Econometrics*, 236(2):105480.

De Paola, M., Scoppa, V., and Falcone, M. (2013). The deterrent effects of the penalty points system for driving offences: a regression discontinuity approach. *Empirical Economics*, 45:965–985.

Deaton, A. (2010). Instruments, randomization, and learning about development. *Journal of economic literature*, 48(2):424–455.

Deshpande, M. and Li, Y. (2019). Who is screened out? application costs and the targeting of disability programs. *American Economic Journal: Economic Policy*, 11(4):213–248.

Diamond, A. and Sekhon, J. S. (2013). Genetic matching for estimating causal effects: A general multivariate matching method for achieving balance in observational studies. *Review of Economics and Statistics*, 95(3):932–945.

DiPrete, T. A. and Gangl, M. (2004). Assessing bias in the estimation of causal effects: Rosenbaum bounds on matching estimators and instrumental variables estimation with imperfect instruments. *Sociological methodology*, 34(1):271–310.

Doan, T. and Iskandar-Datta, M. (2021). Does gender in the c-suite really matter? *Journal of Accounting, Auditing & Finance*, 36(1):81–107.

Doleac, J. L. and Hansen, B. (2020). The unintended consequences of "ban the box": Statistical discrimination and employment outcomes when criminal histories are hidden. *Journal of Labor Economics*, 38(2):321–374.

Dolley, J. C. (1933). Characteristics and procedure of common stock split-ups. *Harvard business review*, 11(3):316–326.

Donald, S. G. and Lang, K. (2007). Inference with difference-in-differences and other panel data. *The review of Economics and Statistics*, 89(2):221–233.

Donoho, D. L. and Johnstone, I. M. (1995). Adapting to unknown smoothness via wavelet shrinkage. *Journal of the american statistical association*, 90(432):1200–1224.

Doornik, J. A. and Hansen, H. (2008). An omnibus test for univariate and multivariate normality. *Oxford bulletin of economics and statistics*, 70:927–939.

Doudchenko, N. and Imbens, G. W. (2016). Balancing, regression, difference-in-differences and synthetic control methods: A synthesis. Technical report, National Bureau of Economic Research.

Driscoll, J. C. and Kraay, A. C. (1998). Consistent covariance matrix estimation with spatially dependent panel data. *Review of economics and statistics*, 80(4):549–560.

Dubow, B. and Monteiro, N. B. (2006). Measuring market cleanliness. *Available at SSRN 1019999*.

Dufour, J.-M. (1997). Some impossibility theorems in econometrics with applications to structural and dynamic models. *Econometrica: Journal of the Econometric Society*, pages 1365–1387.

Dutta, A., Knif, J., Kolari, J. W., and Pynnonen, S. (2018). A robust and powerful test of abnormal stock returns in long-horizon event studies. *Journal of Empirical Finance*, 47:1–24.

Ebbes, P., Papies, D., and Van Heerde, H. J. (2011). The sense and non-sense of holdout sample validation in the presence of endogeneity. *Marketing Science*, 30(6):1115–1122.

Ebbes, P., Wedel, M., Böckenholt, U., and Steerneman, T. (2005). Solving and testing for regressor-error (in) dependence when no instrumental variables are available: With new evidence for the effect of education on income. *Quantitative Marketing and Economics*, 3:365–392.

Eckles, D. and Bakshy, E. (2021). Bias and high-dimensional adjustment in observational studies of peer effects. *Journal of the American Statistical Association*, 116(534):507–517.

Efron, B. (1992). Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics: Methodology and distribution*, pages 569–593. Springer.

Einav, L., Finkelstein, A., and Levin, J. (2010). Beyond testing: Empirical models of insurance markets. *Annu. Rev. Econ.*, 2(1):311–336.

Ekwaru, J. P. and Veugelers, P. J. (2018). The overlooked importance of constants added in log transformation of independent variables with zero values: A proposed approach for determining an optimal constant. *Statistics in Biopharmaceutical Research*, 10(1):26–29.

Elberse, A. (2007). The power of stars: Do star actors drive the success of movies? *Journal of marketing*, 71(4):102–120.

Esarey, J. and Sumner, J. L. (2018). Marginal effects in interaction models: Determining and controlling the false positive rate. *Comparative Political Studies*, 51(9):1144–1176.

Faber, B. and Gaubert, C. (2019). Tourism and economic development: Evidence from mexico's coastline. *American Economic Review*, 109(6):2245–2293.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417.

Fama, E. F., Fisher, L., Jensen, M. C., and Roll, R. (1969). The adjustment of stock prices to new information. *International economic review*, 10(1):1–21.

Fama, E. F. and French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of financial economics*, 33(1):3–56.

Fama, E. F. and French, K. R. (2010). Luck versus skill in the cross-section of mutual fund returns. *The journal of finance*, 65(5):1915–1947.

Fan, Y. and Yu, Z. (2012). Partial identification of distributional and quantile treatment effects in difference-in-differences models. *Economics Letters*, 115(3):511–515.

Fang, E., Lee, J., and Yang, Z. (2015). The timing of codevelopment alliances in new product development processes: Returns for upstream and downstream partners. *Journal of Marketing*, 79(1):64–82.

Faraway, J. J. (2016). *Extending the linear model with R: generalized linear, mixed effects and nonparametric regression models*. CRC press.

Ferman, B. and Pinto, C. (2021). Synthetic controls with imperfect pretreatment fit. *Quantitative Economics*, 12(4):1197–1221.

Firpo, S. and Possebom, V. (2018). Synthetic control method: Inference, sensitivity analysis and confidence sets. *Journal of Causal Inference*, 6(2):20160026.

Fornell, C., Mithas, S., Morgeson III, F. V., and Krishnan, M. S. (2006). Customer satisfaction and stock prices: High returns, low risk. *Journal of marketing*, 70(1):3–14.

Fossum, J. L. and Montoya, A. K. (2023). When to use different inferential methods for power analysis and data analysis for between-subjects mediation. *Advances in Methods and Practices in Psychological Science*, 6(2):25152459231156606.

Fox, J. (1997). *Applied regression analysis, linear models, and related methods*. sage publications, Inc.

Freedman, D. A. (2008). Randomization does not justify logistic regression. *Statistical Science*, pages 237–249.

Fricke, H. (2017). Identification based on difference-in-differences approaches with multiple treatments. *Oxford Bulletin of Economics and Statistics*, 79(3):426–433.

Fritz, M. S., Taylor, A. B., and MacKinnon, D. P. (2012). Explanation of two anomalous results in statistical mediation analysis. *Multivariate Behavioral Research*, 47(1):61–87.

Frölich, M. and Melly, B. (2013). Unconditional quantile treatment effects under endogeneity. *Journal of Business & Economic Statistics*, 31(3):346–357.

Fuchs-Schündeln, N. and Hassan, T. A. (2016). Natural experiments in macroeconomics. In *Handbook of macroeconomics*, volume 2, pages 923–1012. Elsevier.

Furnival, G. M. and Wilson, R. W. (2000). Regressions by leaps and bounds. *Technometrics*, 42(1):69–79.

Gallego, F., Montero, J.-P., and Salas, C. (2013). The effect of transport policies on car use: Evidence from latin american cities. *Journal of Public Economics*, 107:47–62.

Gallen, T. (2020). Broken instruments. *Available at SSRN 3671850*.

Gao, H., Xie, J., Wang, Q., and Wilbur, K. C. (2015). Should ad spending increase or decrease before a recall announcement? the marketing–finance interface in product-harm crisis management. *Journal of Marketing*, 79(5):80–99.

Gardner, J. (2022). Two-stage differences in differences. *arXiv preprint arXiv:2207.05943*.

Gasser, T. and Müller, H.-G. (1979). Kernel estimation of regression functions. In *Smoothing Techniques for Curve Estimation: Proceedings of a Workshop held in Heidelberg, April 2–4, 1979*, pages 23–68. Springer.

Geary, R. C. (1936). Moments of the ratio of the mean deviation to the standard deviation for normal samples. *Biometrika*, pages 295–307.

Gelman, A. and Imbens, G. (2019). Why high-order polynomials should not be used in regression discontinuity designs. *Journal of Business & Economic Statistics*, 37(3):447–456.

Gerard, F., Rokkanen, M., and Rothe, C. (2020). Bounds on treatment effects in regression discontinuity designs with a manipulated running variable. *Quantitative Economics*, 11(3):839–870.

Gerber, A. S., Green, D. P., Kaplan, E. H., and Kern, H. L. (2010). Baseline, placebo, and treatment: Efficient estimation for three-group experiments. *Political Analysis*, 18(3):297–315.

Geyskens, I., Gielens, K., and Dekimpe, M. G. (2002). The market valuation of internet channel additions. *Journal of marketing*, 66(2):102–119.

Gibbons, C. E., Suárez Serrato, J. C., and Urbancic, M. B. (2018). Broken or fixed effects? *Journal of Econometric Methods*, 8(1):20170002.

Gielens, K., Van de Gucht, L. M., Steenkamp, J.-B. E., and Dekimpe, M. G. (2008). Dancing with a giant: The effect of wal-mart's entry into the united kingdom on the performance of european retailers. *Journal of Marketing Research*, 45(5):519–534.

Glasser, M. (1964). Linear regression analysis with missing observations among the independent variables. *Journal of the American Statistical Association*, 59(307):834–844.

Glejser, H. (1969). A new test for heteroskedasticity. *Journal of the American Statistical Association*, 64(325):316–323.

Gobillon, L. and Magnac, T. (2016). Regional policy evaluation: Interactive fixed effects and synthetic controls. *Review of Economics and Statistics*, 98(3):535–551.

Godfrey, L. G. (1978). Testing against general autoregressive and moving average error models when the regressors include lagged dependent variables. *Econometrica: Journal of the Econometric Society*, pages 1293–1301.

Goldberger, A. S. (1991). *A course in econometrics*. Harvard University Press.

Goldfarb, A. and Tucker, C. (2011). Online display advertising: Targeting and obtrusiveness. *Marketing Science*, 30(3):389–404.

Goldfarb, A., Tucker, C., and Wang, Y. (2022). Conducting research in marketing with quasi-experiments. *Journal of Marketing*, 86(3):1–20.

Goldfeld, S. M. and Quandt, R. E. (1965). Some tests for homoscedasticity. *Journal of the American statistical Association*, 60(310):539–547.

Goldsmith-Pinkham, P., Hull, P., and Kolesár, M. (2022). Contamination bias in linear regressions. Technical report, National Bureau of Economic Research.

Goodman-Bacon, A. (2021). Difference-in-differences with variation in treatment timing. *Journal of Econometrics*, 225(2):254–277.

Gordon, B. R., Zettelmeyer, F., Bhargava, N., and Chapsky, D. (2019). A comparison of approaches to advertising measurement: Evidence from big field experiments at facebook. *Marketing Science*, 38(2):193–225.

Gormley, T. A. and Matsa, D. A. (2011). Growing out of trouble? corporate responses to liability risk. *The Review of Financial Studies*, 24(8):2781–2821.

Gould, W., Pitblado, J., and Poi, B. (2010). Maximum likelihood estimation with stata: Statacorp lp.

Gourieroux, C., Holly, A., and Monfort, A. (1982). Likelihood ratio test, wald test, and kuhn-tucker test in linear models with inequality constraints on the regression parameters. *Econometrica: journal of the Econometric Society*, pages 63–80.

Greene, W. H. (1990). A gamma-distributed stochastic frontier model. *Journal of econometrics*, 46(1-2):141–163.

Greevy, R., Lu, B., Silber, J. H., and Rosenbaum, P. (2004). Optimal multivariate matching before randomization. *Biostatistics*, 5(2):263–275.

Guo, T., Sriram, S., and Manchanda, P. (2020). "let the sunshine in": The impact of industry payment disclosure on physician prescription behavior. *Marketing Science*, 39(3):516–539.

Habel, J., Alavi, S., and Linsenmayer, K. (2021). Variable compensation and salesperson health. *Journal of Marketing*, 85(3):130–149.

Hainmueller, J. (2012). Entropy balancing for causal effects: A multivariate reweighting method to produce balanced samples in observational studies. *Political analysis*, 20(1):25–46.

Haitovsky, Y. (1968). Missing data in regression analysis. *Journal of the Royal Statistical Society: Series B (Methodological)*, 30(1):67–82.

Hall, A. R., Rudebusch, G. D., and Wilcox, D. W. (1996). Judging instrument relevance in instrumental variables estimation. *International Economic Review*, pages 283–298.

Hall, G. C., Hutchinson, P. J., and Michaelas, N. (2004). Determinants of the capital structures of european smes. *Journal of Business Finance & Accounting*, 31(5-6):711–728.

Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., and Stahel, W. A. (2005). Robust statistics. *Wiley Series in Probability and Statistics*.

Hansen, B. B. (2008). The prognostic analogue of the propensity score. *Biometrika*, 95(2):481–488.

Hansen, B. B., Rosenbaum, P. R., and Small, D. S. (2014). Clustered treatment assignments and sensitivity to unmeasured biases in observational studies. *Journal of the American Statistical Association*, 109(505):133–144.

Hansen, L. P. (1982). Large sample properties of generalized method of moments estimators. *Econometrica: Journal of the econometric society*, pages 1029–1054.

Harper, S. and Bruckner, T. A. (2017). Did the great recession increase suicides in the usa? evidence from an interrupted time-series analysis. *Annals of epidemiology*, 27(7):409–414.

Hartmann, W., Nair, H. S., and Narayanan, S. (2011). Identifying causal marketing mix effects using a regression discontinuity design. *Marketing Science*, 30(6):1079–1097.

Harvey, A. C. and Collier, P. (1977). Testing for functional misspecification in regression analysis. *Journal of Econometrics*, 6(1):103–119.

Hastie, T., Mazumder, R., Lee, J. D., and Zadeh, R. (2015). Matrix completion and low-rank svd via fast alternating least squares. *The Journal of Machine Learning Research*, 16(1):3367–3402.

Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.

Hausman, C. and Rapson, D. S. (2018). Regression discontinuity in time: Considerations for empirical applications. *Annual Review of Resource Economics*, 10:533–552.

Hausman, J. A. (1978). Specification tests in econometrics. *Econometrica: Journal of the econometric society*, pages 1251–1271.

Hausman, J. A. (1983). Specification and estimation of simultaneous equation models. *Handbook of econometrics*, 1:391–448.

Hayes, A. F. and Preacher, K. J. (2014). Statistical mediation analysis with a multicategorical independent variable. *British journal of mathematical and statistical psychology*, 67(3):451–470.

Hayes, A. F. and Scharkow, M. (2013). The relative trustworthiness of inferential tests of the indirect effect in statistical mediation analysis: does method really matter? *Psychological science*, 24(10):1918–1927.

He, C., Ozturk, O. C., Gu, C., and Silva-Risso, J. M. (2021). The end of the express road for hybrid vehicles: Can governments' green product incentives backfire? *Marketing Science*, 40(1):80–100.

He, S., Hollenbeck, B., and Proserpio, D. (2022). The market for fake reviews. *Marketing Science*, 41(5):896–921.

Heath, D., Ringgenberg, M. C., Samadi, M., and Werner, I. M. (2023). Reusing natural experiments. *The Journal of Finance*, 78(4):2329–2364.

Heckman, J. (1990). Varieties of selection bias. *The American Economic Review*, 80(2):313–318.

Heckman, J. and Navarro-Lozano, S. (2004). Using matching, instrumental variables, and control functions to estimate economic choice models. *Review of Economics and statistics*, 86(1):30–57.

Heckman, J. J. (1976). The common structure of statistical models of truncation, sample selection and limited dependent variables and a simple estimator for such models. In *Annals of economic and social measurement, volume 5, number 4*, pages 475–492. NBER.

Heckman, J. J., Ichimura, H., and Todd, P. E. (1997). Matching as an econometric evaluation estimator: Evidence from evaluating a job training programme. *The review of economic studies*, 64(4):605–654.

Heckman, J. J., LaLonde, R. J., and Smith, J. A. (1999). The economics and econometrics of active labor market programs. In *Handbook of labor economics*, volume 3, pages 1865–2097. Elsevier.

Heckman, J. J. and Smith, J. A. (1999). The pre-programme earnings dip and the determinants of participation in a social programme. implications for simple programme evaluation strategies. *The Economic Journal*, 109(457):313–348.

Heller, R., Rosenbaum, P. R., and Small, D. S. (2009). Split samples and design sensitivity in observational studies. *Journal of the American Statistical Association*, 104(487):1090–1101.

Henderson, C. R. (1975). Best linear unbiased estimation and prediction under a selection model. *Biometrics*, pages 423–447.

Hill, A. D., Johnson, S. G., Greco, L. M., O'Boyle, E. H., and Walter, S. L. (2021). Endogeneity: A review and agenda for the methodology-practice divide affecting micro and macro research. *Journal of Management*, 47(1):105–143.

Hirtle, B., Kovner, A., and Plosser, M. (2020). The impact of supervision on bank performance. *The Journal of Finance*, 75(5):2765–2808.

Hjort, J. and Poulsen, J. (2019). The arrival of fast internet and employment in africa. *American Economic Review*, 109(3):1032–1079.

Ho, D. E., Imai, K., King, G., and Stuart, E. A. (2007). Matching as nonparametric preprocessing for reducing model dependence in parametric causal inference. *Political analysis*, 15(3):199–236.

Hochberg, Y. (1988). A sharper bonferroni procedure for multiple tests of significance. *Biometrika*, 75(4):800–802.

Hodges Jr, J. L. and Lehmann, E. L. (2011). Estimates of location based on rank tests. In *Selected Works of EL Lehmann*, pages 287–300. Springer.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Hollander, M., Wolfe, D. A., and Chicken, E. (2013). *Nonparametric statistical methods*. John Wiley & Sons.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70.

Homburg, C., Vollmayr, J., and Hahn, A. (2014). Firm value creation through major channel expansions: Evidence from an event study in the united states, germany, and china. *Journal of Marketing*, 78(3):38–61.

Honda, Y. (1985). Testing the error components model with non-normal disturbances. *The review of economic studies*, 52(4):681–690.

Horsky, D. and Swyngedouw, P. (1987). Does it pay to change your company's name? a stock market perspective. *Marketing Science*, 6(4):320–335.

Hosmer, D. W. and Lemeshow, S. (1992). Confidence interval estimation of interaction. *Epidemiology*, pages 452–456.

Houston, M. B. and Johnson, S. A. (2000). Buyer–supplier contracts versus joint ventures: Determinants and consequences of transaction structure. *Journal of marketing research*, 37(1):1–15.

Hoyer, P., Janzing, D., Mooij, J. M., Peters, J., and Schölkopf, B. (2008). Nonlinear causal discovery with additive noise models. *Advances in neural information processing systems*, 21.

Hoyle, R. H. (1999). *Statistical strategies for small sample research*. sage.

Hoynes, H. W. and Schanzenbach, D. W. (2009). Consumption responses to in-kind transfers: Evidence from the introduction of the food stamp program. *American Economic Journal: Applied Economics*, 1(4):109–139.

Huber, M., Schelker, M., and Strittmatter, A. (2022). Direct and indirect effects based on changes-in-changes. *Journal of Business & Economic Statistics*, 40(1):432–443.

Huber, P. J. (1964). Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101.

Hurvich, C. M. and Tsai, C.-L. (1989). Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307.

Iacus, S. M., King, G., and Porro, G. (2012). Causal inference without balance checking: Coarsened exact matching. *Political analysis*, 20(1):1–24.

Imai, K., Jiang, Z., and Malani, A. (2021a). Causal inference with interference and noncompliance in two-stage randomized experiments. *Journal of the American Statistical Association*, 116(534):632–644.

Imai, K., Keele, L., and Tingley, D. (2010a). A general approach to causal mediation analysis. *Psychological methods*, 15(4):309.

Imai, K., Keele, L., and Yamamoto, T. (2010b). Identification, inference and sensitivity analysis for causal mediation effects.

Imai, K. and Kim, I. S. (2019). When should we use unit fixed effects regression models for causal inference with longitudinal data? *American Journal of Political Science*, 63(2):467–490.

Imai, K. and Kim, I. S. (2021). On the use of two-way fixed effects regression models for causal inference with panel data. *Political Analysis*, 29(3):405–415.

Imai, K., Kim, I. S., and Wang, E. H. (2021b). Matching methods for causal inference with time-series cross-sectional data. *American Journal of Political Science*.

Imai, K., King, G., and Stuart, E. A. (2008). Misunderstandings between experimentalists and observationalists about causal inference. *Journal of the royal statistical society: series A (statistics in society)*, 171(2):481–502.

Imbens, G. and Kalyanaraman, K. (2012). Optimal bandwidth choice for the regression discontinuity estimator. *The Review of economic studies*, 79(3):933–959.

Imbens, G. and Lemieux, T. (2008). The regression discontinuity design–theory and applications. *Journal of Econometrics*.

Imbens, G. W. and Angrist, J. D. (1994). Identification and estimation of local average treatment effects. *Econometrica*, 62(2):467–475.

Imbens, G. W. and Rubin, D. B. (2015). *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press.

Iskandar-Datta, M. and Jia, Y. (2013). Valuation consequences of clawback provisions. *The Accounting Review*, 88(1):171–198.

Israeli, A. (2018). Online map enforcement: evidence from a quasi-experiment. *Marketing Science*, 37(5):710–732.

Ittner, C., Larcker, D., and Taylor, D. (2009). Commentary–the stock market's pricing of customer satisfaction. *Marketing Science*, 28(5):826–835.

Jacobson, R. and Mizik, N. (2009). The financial markets and customer satisfaction: Reexamining possible financial market mispricing of customer satisfaction. *Marketing Science*, 28(5):810–819.

Jaffe, J. F. (1974). Special information and insider trading. *The Journal of Business*, 47(3):410–428.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer.

Janakiraman, R., Lim, J. H., and Rishika, R. (2018). The effect of a data breach announcement on customer behavior: Evidence from a multichannel retailer. *Journal of marketing*, 82(2):85–105.

Jepsen, C., Troske, K., and Coomes, P. (2014). The labor-market returns to community college degrees, diplomas, and certificates. *Journal of Labor Economics*, 32(1):95–121.

Jin, H. and Rubin, D. B. (2008). Principal stratification for causal inference with extended partial compliance. *Journal of the American Statistical Association*, 103(481):101–111.

Jin, H. and Rubin, D. B. (2009). Public schools versus private schools: Causal inference with partial compliance. *Journal of Educational and Behavioral Statistics*, 34(1):24–45.

Johansson, P. and Schultzberg, M. (2022). Rerandomization: A complement or substitute for stratification in randomized experiments? *Journal of Statistical Planning and Inference*, 218:43–58.

Johnson, M. S. (2020). Regulation by shaming: Deterrence effects of publicizing violations of workplace safety and health laws. *American economic review*, 110(6):1866–1904.

Johnson, N. L. (1949). Systems of frequency curves generated by methods of translation. *Biometrika*, 36(1/2):149.

Johnson, P. O. and Neyman, J. (1936). Tests of certain linear hypotheses and their application to some educational problems. *Statistical research memoirs*.

Johnson, R. A., Wichern, D. W., et al. (2002). Applied multivariate statistical analysis.

Johnston, M. A. (2007). A review of the application of event studies in marketing. *Academy of Marketing Science Review*, 2007:1.

Jones, M. P. (1996). Indicator and stratification methods for missing explanatory variables in multiple linear regression. *Journal of the American statistical association*, 91(433):222–230.

Jordon, J., Szpruch, L., Houssiau, F., Bottarelli, M., Cherubin, G., Maple, C., Cohen, S. N., and Weller, A. (2022). Synthetic data–what, why and how? *arXiv preprint arXiv:2205.03257*.

Joshi, A. M. and Hanssens, D. M. (2009). Movie advertising and the stock market valuation of studios: a case of "great expectations?". *Marketing Science*, 28(2):239–250.

Kahn-Lang, A. and Lang, K. (2020). The promise and pitfalls of differences-in-differences: Reflections on 16 and pregnant and other applications. *Journal of Business & Economic Statistics*, 38(3):613–620.

Kalaignanam, K. and Bahadir, S. C. (2013). Corporate brand name changes and business restructuring: is the relationship complementary or substitutive? *Journal of the Academy of Marketing Science*, 41:456–472.

Kapelner, A. and Krieger, A. (2014). Matching on-the-fly: Sequential allocation with higher power and efficiency. *Biometrics*, 70(2):378–388.

Karniouchina, E. V., Moore, W. L., and Cooney, K. J. (2009). Impact of mad money stock recommendations: merging financial and marketing perspectives. *Journal of Marketing*, 73(6):244–266.

Karniouchina, E. V., Uslay, C., and Erenburg, G. (2011). Do marketing media have life cycles? the case of product placement in movies. *Journal of Marketing*, 75(3):27–48.

Keele, L. (2010). An overview of rbounds: An r package for rosenbaum bounds sensitivity analysis with matched data. *White Paper. Columbus, OH*, 1:15.

Keele, L. and Grieve, R. (2025). So many choices: A guide to selecting among methods to adjust for observed confounders. *Statistics in Medicine*, 44(5):e10336.

Keller, K. O., Guyt, J. Y., and Grewal, R. (2024). Soda taxes and marketing conduct. *Journal of Marketing Research*, 61(3):393–410.

Kenny, D. A. and Judd, C. M. (2014). Power anomalies in testing mediation. *Psychological science*, 25(2):334–339.

Kim, J.-S. and Frees, E. W. (2007). Multilevel modeling with correlated effects. *Psychometrika*, 72(4):505–533.

Kim, M. and McAlister, L. M. (2011). Stock market reaction to unexpected growth in marketing expenditure: Negative for sales force, contingent on spending level for advertising. *Journal of Marketing*, 75(4):68–85.

Kim, S., Lee, C., and Gupta, S. (2020). Bayesian synthetic control methods. *Journal of Marketing Research*, 57(5):831–852.

King, G. (1995). Replication, replication. *PS: Political Science & Politics*, 28(3):444–452.

King, G., Lucas, C., and Nielsen, R. A. (2017). The balance-sample size frontier in matching methods for causal inference. *American journal of political science*, 61(2):473–489.

King, G. and Nielsen, R. (2019). Why propensity scores should not be used for matching. *Political analysis*, 27(4):435–454.

King, M. L. and Wu, P. X. (1997). Locally optimal one-sided tests for multiparameter hypotheses. *Econometric Reviews*, 16(2):131–156.

Kiviet, J. F. (2020). Testing the impossible: Identifying exclusion restrictions. *Journal of Econometrics*, 218(2):294–316.

Kleinberg, J., Ludwig, J., Mullainathan, S., and Obermeyer, Z. (2015). Prediction policy problems. *American Economic Review*, 105(5):491–495.

Kleven, H. J. (2016). Bunching. *Annual Review of Economics*, 8:435–464.

Knol, M. J. and VanderWeele, T. J. (2012). Recommendations for presenting analyses of effect modification and interaction. *International journal of epidemiology*, 41(2):514–520.

Kothari, S. and Warner, J. B. (1997). Measuring long-horizon security price performance. *Journal of financial economics*, 43(3):301–339.

Krauth, B. (2016). Bounding a linear causal effect using relative correlation restrictions. *Journal of Econometric Methods*, 5(1):117–141.

Laird, N. M. and Ware, J. H. (1982). Random-effects models for longitudinal data. *Biometrics*, pages 963–974.

LaLonde, R. J. (1986). Evaluating the econometric evaluations of training programs with experimental data. *The American economic review*, pages 604–620.

Lambrecht, A., Tucker, C., and Zhang, X. (2024). Tv advertising and online sales: A case study of intertemporal substitution effects for an online travel platform. *Journal of Marketing Research*, 61(2):248–270.

Lancaster, T. (2000). The incidental parameter problem since 1948. *Journal of econometrics*, 95(2):391–413.

Lane, V. and Jacobson, R. (1995). Stock market reactions to brand extension announcements: The effects of brand attitude and familiarity. *Journal of marketing*, 59(1):63–77.

Larcker, D. F. and Rusticus, T. O. (2010). On the use of instrumental variables in accounting research. *Journal of accounting and economics*, 49(3):186–205.

Lavy, V., Angrist, J. D., and Schlosser, A. (2006). New evidence on the causal link between the quantity and quality of children.

Ledgerwood, A. and Shrout, P. E. (2011). The trade-off between accuracy and precision in latent variable models of mediation processes. *Journal of personality and social psychology*, 101(6):1174.

Lee, D. S. (2009). Training, wages, and sample selection: Estimating sharp bounds on treatment effects. *The Review of Economic Studies*, pages 1071–1102.

Lee, D. S. and Lemieux, T. (2010). Regression discontinuity designs in economics. *Journal of economic literature*, 48(2):281–355.

Lee, D. S., McCrary, J., Moreira, M. J., and Porter, J. (2022). Valid t-ratio inference for iv. *American Economic Review*, 112(10):3260–3290.

Lee Rodgers, J., Beasley, W. H., and Schuelke, M. (2014). Graphical data analysis on the circle: Wrap-around time series plots for (interrupted) time series designs. *Multivariate Behavioral Research*, 49(6):571–580.

Lewbel, A. (1997). Constructing instruments for regressions with measurement error when no additional data are available, with an application to patents and r&d. *Econometrica: journal of the econometric society*, pages 1201–1213.

Lewbel, A. (2012). Using heteroscedasticity to identify and estimate mismeasured and endogenous regressor models. *Journal of Business & Economic Statistics*, 30(1):67–80.

Li, H., Liu, X., Chen, Y., and Fan, Y. (2022). Testing for serial correlation in autoregressive exogenous models with possible garch errors. *Entropy*, 24(8):1076.

Li, K. T. (2024). Frontiers: A simple forward difference-in-differences method. *Marketing Science*, 43(2):267–279.

Li, K. T. and Sonnier, G. P. (2023). Statistical inference for the factor model approach to estimate causal effects in quasi-experimental settings. *Journal of Marketing Research*, 60(3):449–472.

Li, K. T. and Van den Bulte, C. (2023). Augmented difference-in-differences. *Marketing Science*, 42(4):746–767.

Li, S., Vlassis, N., Kawale, J., and Fu, Y. (2016). Matching via dimensionality reduction for estimation of treatment effects in digital marketing campaigns. In *IJCAI*, volume 16, pages 3768–3774.

Li, X., Gan, C., and Hu, B. (2011). The welfare impact of microcredit on rural households in china. *The Journal of Socio-Economics*, 40(4):404–411.

Li, Y. P., Propert, K. J., and Rosenbaum, P. R. (2001). Balanced risk set matching. *Journal of the American Statistical Association*, 96(455):870–882.

Liang, K.-Y. and Zeger, S. L. (1986). Longitudinal data analysis using generalized linear models. *Biometrika*, 73(1):13–22.

Liaukonyte, J., Teixeira, T., and Wilbur, K. C. (2015). Television advertising and online shopping. *Marketing Science*, 34(3):311–330.

Liaukonytė, J., Tuchman, A., and Zhu, X. (2023). Frontiers: Spilling the beans on political consumerism: Do social media boycotts and buycotts translate to real sales impact? *Marketing Science*, 42(1):11–25.

Lim, J. H., Rishika, R., Janakiraman, R., and Kannan, P. (2020). Competitive effects of front-of-package nutrition labeling adoption on nutritional quality: evidence from facts up front–style labels. *Journal of Marketing*, 84(6):3–21.

Linden, A. (2015). Conducting interrupted time-series analysis for single-and multiple-group comparisons. *The Stata Journal*, 15(2):480–500.

Linden, A. (2017). A comprehensive set of postestimation measures to enrich interrupted time-series analysis. *The Stata Journal*, 17(1):73–88.

Linden, A. and Yarnold, P. R. (2016). Using machine learning to identify structural breaks in single-group interrupted time series designs. *Journal of Evaluation in Clinical Practice*, 22(6):855–859.

Little, R. J. (1988). A test of missing completely at random for multivariate data with missing values. *Journal of the American statistical Association*, 83(404):1198–1202.

Little, R. J. (1992). Regression with missing x's: a review. *Journal of the American statistical association*, 87(420):1227–1237.

Liu, L., Wang, Y., and Xu, Y. (2022). A practical guide to counterfactual estimators for causal inference with time-series cross-sectional data. *American Journal of Political Science*.

Ljung, G. M. and Box, G. E. (1978). On a measure of lack of fit in time series models. *Biometrika*, 65(2):297–303.

Long, Q., Little, R. J., and Lin, X. (2010). Estimating causal effects in trials involving multitreatment arms subject to non-compliance: a bayesian framework. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 59(3):513–531.

Looney, S. W. and Gulledge Jr, T. R. (1985). Use of the correlation coefficient with normal probability plots. *The American Statistician*, 39(1):75–79.

Lopez, M. J. and Gutman, R. (2017). Estimation of causal effects with multiple treatments: a review and new ideas. *Statistical Science*, pages 432–454.

Loughran, T. and Ritter, J. R. (1995). The new issues puzzle. *The Journal of finance*, 50(1):23–51.

Loughran, T. and Ritter, J. R. (2000). Uniformly least powerful tests of market efficiency. *Journal of financial economics*, 55(3):361–389.

Lovell, M. C. (2008). A simple proof of the fwl theorem. *The Journal of Economic Education*, 39(1):88–91.

Lyon, J. D., Barber, B. M., and Tsai, C.-L. (1999). Improved methods for tests of long-run abnormal stock returns. *The Journal of Finance*, 54(1):165–201.

MacKinlay, A. C. (1997). Event studies in economics and finance. *Journal of economic literature*, 35(1):13–39.

MacKinnon, D. P., Fairchild, A. J., and Fritz, M. S. (2007). Mediation analysis. *Annu. Rev. Psychol.*, 58:593–614.

MacKinnon, D. P., Warsi, G., and Dwyer, J. H. (1995). A simulation study of mediated effect measures. *Multivariate behavioral research*, 30(1):41–62.

MacKinnon, J. G. and White, H. (1985). Some heteroskedasticity-consistent covariance matrix estimators with improved finite sample properties. *Journal of econometrics*, 29(3):305–325.

Magel, R. C. and Hertsgaard, D. (1987). A collinearity diagnostic for nonlinear regression: A collinearity diagnostic. *Communications in Statistics-Simulation and Computation*, 16(1):85–97.

Malhotra, A. and Kubowicz Malhotra, C. (2011). Evaluating customer information breaches as service failures: An event study approach. *Journal of Service Research*, 14(1):44–59.

Mallows, C. L. (1995). More comments on cp. *Technometrics*, 37(4):362–372.

Manchanda, P., Packard, G., and Pattabhiramaiah, A. (2015). Social dollars: The economic impact of customer participation in a firm-sponsored online customer community. *Marketing Science*, 34(3):367–387.

Manly, B. F. (1976). Exponential data transformations. *Journal of the Royal Statistical Society Series D: The Statistician*, 25(1):37–42.

Marcus, M. and Sant'Anna, P. H. (2021). The role of parallel trends in event study settings: An application to environmental economics. *Journal of the Association of Environmental and Resource Economists*, 8(2):235–275.

Mardia, K. V. (1970). Measures of multivariate skewness and kurtosis with applications. *Biometrika*, 57(3):519–530.

Markovitch, D. G. and Golder, P. N. (2008). Findings—using stock prices to predict market events: Evidence on sales takeoff and long-term firm survival. *Marketing Science*, 27(4):717–729.

Marsaglia, G. and Marsaglia, J. (2004). Evaluating the anderson-darling distribution. *Journal of statistical software*, 9:1–5.

Masten, M. A. and Poirier, A. (2022). The effect of omitted variables on the sign of regression coefficients. *arXiv preprint arXiv:2208.00552*.

Masulis, R. W. and Nahata, R. (2011). Venture capital conflicts of interest: Evidence from acquisitions of venture-backed firms. *Journal of Financial and Quantitative Analysis*, 46(2):395–430.

Mazodier, M. and Rezaee, A. (2013). Are sponsorship announcements good news for the shareholders? evidence from international stock exchanges. *Journal of the Academy of Marketing Science*, 41:586–600.

Mazumder, R., Hastie, T., and Tibshirani, R. (2010). Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11:2287–2322.

McCaffrey, D. F., Griffin, B. A., Almirall, D., Slaughter, M. E., Ramchand, R., and Burgette, L. F. (2013). A tutorial on propensity score estimation for multiple treatments using generalized boosted models. *Statistics in medicine*, 32(19):3388–3414.

McCrary, J. (2008). Manipulation of the running variable in the regression discontinuity design: A density test. *Journal of econometrics*, 142(2):698–714.

McCullagh, P. (2019). *Generalized linear models*. Routledge.

McWilliams, A. and Siegel, D. (1997). Event studies in management research: Theoretical and empirical issues. *Academy of management journal*, 40(3):626–657.

Mei, B. and Sun, C. (2008). Event analysis of the impact of mergers and acquisitions on the financial performance of the u.s. forest products industry. *Forest Policy and Economics*, 10(5):286–294.

Mellon, J. (2023). Rain, rain, go away: 195 potential exclusion-restriction violations for studies using weather as an instrumental variable. *Available at SSRN 3715610*.

Miller, R. G. (1974). The jackknife-a review. *Biometrika*, 61(1):1–15.

Mirenda, L., Mocetti, S., and Rizzica, L. (2022). The economic effects of mafia: Firm level evidence. *American Economic Review*, 112(8):2748–2773.

Mitchell, M. L. and Stafford, E. (2000). Managerial decisions and long-term stock price performance. *The Journal of Business*, 73(3):287–329.

Moon, H. R. and Weidner, M. (2015). Linear regression for panel with unknown number of factors as interactive fixed effects. *Econometrica*, 83(4):1543–1579.

Moreira, H. and Moreira, M. J. (2019). Optimal two-sided tests for instrumental variables regression with heteroskedastic and autocorrelated errors. *Journal of Econometrics*, 213(2):398–433.

Moreira, M. J. (2003). A conditional likelihood ratio test for structural models. *Econometrica*, 71(4):1027–1048.

Moreira, M. J. (2009). Tests with correct size when instruments can be arbitrarily weak. *Journal of Econometrics*, 152(2):131–140.

Morgan, K. L. and Rubin, D. B. (2012). Rerandomization to improve covariate balance in experiments.

Mroz, T. A. (1984). *The sensitivity of an empirical model of married women's hours of work to economic and statistical assumptions*. Stanford University.

Mullainathan, S. and Spiess, J. (2017). Machine learning: an applied econometric approach. *Journal of Economic Perspectives*, 31(2):87–106.

Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142.

Narayanan, S. and Kalyanam, K. (2015). Position effects in search advertising and their moderators: A regression discontinuity approach. *Marketing Science*, 34(3):388–407.

Narayanan, S. and Nair, H. S. (2013). Estimating causal installed-base effects: A bias-correction approach. *Journal of Marketing Research*, 50(1):70–94.

Nelder, J. A. and Wedderburn, R. W. (1972). Generalized linear models. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 135(3):370–384.

Nelson, C. and Startz, R. (1988). The distribution of the instrumental variables estimator and its t-ratiowhen the instrument is a poor one.

Nelson, C. R. and Startz, R. (1990). The distribution of the instrumental variables estimator and its t-ratio when the instrument is a poor one. *Journal of Business*, pages S125–S140.

Nerlove, M. (1971). Further evidence on the estimation of dynamic economic relations from a time series of cross sections. *Econometrica: Journal of the Econometric Society*, pages 359–382.

Newey, W. K. and McFadden, D. (1994). Large sample estimation and hypothesis testing. *Handbook of econometrics*, 4:2111–2245.

Newey, W. K. and West, K. D. (1986). A simple, positive semi-definite, heteroskedasticity and autocorrelationconsistent covariance matrix.

Nickell, S. (1981). Biases in dynamic models with fixed effects. *Econometrica: Journal of the econometric society*, pages 1417–1426.

Nielsen, H. S., Sørensen, T., and Taber, C. (2010). Estimating the effect of student aid on college enrollment: Evidence from a government grant policy reform. *American Economic Journal: Economic Policy*, 2(2):185–215.

Norris, S., Pecenco, M., and Weaver, J. (2021). The effects of parental and sibling incarceration: Evidence from ohio. *American Economic Review*, 111(9):2926–2963.

Oestreicher-Singer, G. and Zalmanson, L. (2013). Content or community? a digital business strategy for content providers in the social age. *MIS quarterly*, pages 591–616.

O'Hara, R. and Kotze, J. (2010). Do not log-transform count data. *Nature Precedings*, pages 1–1.

Oster, E. (2019). Unobservable selection and coefficient stability: Theory and evidence. *Journal of Business & Economic Statistics*, 37(2):187–204.

Owen, J. and Rabinovitch, R. (1983). On the class of elliptical distributions and their applications to the theory of portfolio choice. *The Journal of Finance*, 38(3):745–752.

Pandey, V. K., Shanahan, K. J., and Hansen, S. W. (2005). The relationship between shareholder wealth effects, diversity, and publicity as a marketing strategy. *Journal of the Academy of Marketing Science*, 33:423–431.

Pang, X., Liu, L., and Xu, Y. (2022). A bayesian alternative to synthetic control for comparative case studies. *Political Analysis*, 30(2):269–288.

Parente, P. M. and Silva, J. S. (2012). A cautionary note on tests of overidentifying restrictions. *Economics Letters*, 115(2):314–317.

Park, R. E. (1966). Estimation with heteroscedastic error terms. *Econometrica*, 34(4).

Park, S. and Gupta, S. (2012). Handling endogenous regressors by joint estimation using copulas. *Marketing Science*, 31(4):567–586.

Patell, J. M. (1976). Corporate forecasts of earnings per share and stock price behavior: Empirical test. *Journal of accounting research*, pages 246–276.

Pattabhiramaiah, A., Sriram, S., and Manchanda, P. (2019). Paywalls: Monetizing online content. *Journal of marketing*, 83(2):19–36.

Pearl, J. (2014). Graphical models for probabilistic and causal reasoning. *Computing Handbook, 3rd ed.(1)*, pages 44–1.

Pearl, J. (2019). The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM*, 62(3):54–60.

Pearl, J. and Mackenzie, D. (2018). *The book of why: the new science of cause and effect*. Basic books.

Penfold, R. B. and Zhang, F. (2013). Use of interrupted time series analysis in evaluating health care quality improvements. *Academic pediatrics*, 13(6):S38–S44.

Permutt, T. and Hebel, J. R. (1989). Simultaneous-equation estimation in a clinical trial of the effect of smoking on birth weight. *Biometrics*, pages 619–622.

Petrin, A. and Train, K. (2010). A control function approach to endogeneity in consumer choice models. *Journal of marketing research*, 47(1):3–13.

Peukert, C., Bechtold, S., Batikas, M., and Kretschmer, T. (2022). Regulatory spillovers and data governance: Evidence from the gdpr. *Marketing Science*, 41(4):746–768.

Pitt, M. M. (2011). Overidentification tests and causality: a second response to roodman and morduch. *Access at: http://www. brown. edu/research/projects/pitt*.

Pocock, S. J. and Simon, R. (1975). Sequential treatment assignment with balancing for prognostic factors in the controlled clinical trial. *Biometrics*, pages 103–115.

Porreca, Z. (2022). Synthetic difference-in-differences estimation with staggered treatment timing. *Economics Letters*, 220:110874.

Powers, D. E. and Swinton, S. S. (1984). Effects of self-study for coachable test item types. *Journal of Educational Psychology*, 76(2):266.

Preacher, K. J. and Hayes, A. F. (2004). Spss and sas procedures for estimating indirect effects in simple mediation models. *Behavior research methods, instruments, & computers*, 36:717–731.

Priestley, M. B. and Chao, M. (1972). Non-parametric function fitting. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 34(3):385–392.

Proserpio, D. and Zervas, G. (2017a). Online reputation management: Estimating the impact of management responses on consumer reviews. *Marketing Science*, 36(5):645–665.

Proserpio, D. and Zervas, G. (2017b). Online reputation management: Estimating the impact of management responses on consumer reviews. *Marketing Science*, 36(5):645–665.

Puhani, P. A. (2012). The treatment effect, the cross difference, and the interaction term in nonlinear "difference-in-differences" models. *Economics Letters*, 115(1):85–87.

Qu, Y. and Lipkovich, I. (2009). Propensity score estimation with missing values using a multiple imputation missingness pattern (mimp) approach. *Statistics in medicine*, 28(9):1402–1414.

Raassens, N., Wuyts, S., and Geyskens, I. (2012). The market valuation of outsourcing new product development. *Journal of Marketing Research*, 49(5):682–695.

Ramachandra, V. (2018). Deep learning for causal inference. *arXiv preprint arXiv:1803.00149*.

Ramani, N. and Srinivasan, R. (2019). Effects of liberalization on incumbent firms' marketing-mix responses and performance: evidence from a quasi-experiment. *Journal of marketing*, 83(5):97–114.

Rambachan, A. and Roth, J. (2023). A more credible approach to parallel trends. *Review of Economic Studies*, page rdad018.

Ramsey, J. B. (1969). Tests for specification errors in classical linear least-squares regression analysis. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 31(2):350–371.

Rao, C. R. (1992). Information and the accuracy attainable in the estimation of statistical parameters. In *Breakthroughs in Statistics: Foundations and basic theory*, pages 235–247. Springer.

Rao, R. S., Chandy, R. K., and Prabhu, J. C. (2008). The fruits of legitimacy: Why some new ventures gain more from innovation than others. *Journal of Marketing*, 72(4):58–75.

Reed, W. R. (2015). On the practice of lagging variables to avoid simultaneity. *Oxford Bulletin of Economics and Statistics*, 77(6):897–905.

Reiss, P. C. (2011). Structural workshop paper—descriptive, structural, and experimental empirical methods in marketing research. *Marketing Science*, 30(6):950–964.

Ritter, J. R. (1991). The long-run performance of initial public offerings. *The journal of finance*, 46(1):3–27.

Robins, J. M., Hernan, M. A., and Brumback, B. (2000). Marginal structural models and causal inference in epidemiology. *Epidemiology*, pages 550–560.

Rodgers, J. L., John, C. A. S., and Coleman, R. (2005). Did fertility go up after the oklahoma city bombing? an analysis of births in metropolitan counties in oklahoma, 1990–1999. *Demography*, 42:675–692.

Romano, J. P. and Wolf, M. (2005). Stepwise multiple testing as formalized data snooping. *Econometrica*, 73(4):1237–1282.

Romano, J. P. and Wolf, M. (2016). Efficient computation of adjusted p-values for resampling-based stepdown multiple testing. *Statistics & Probability Letters*, 113:38–40.

Rosenbaum, P. R. (2002). Attributing effects to treatment in matched observational studies. *Journal of the American statistical Association*, 97(457):183–192.

Rosenbaum, P. R. (2015). Two r packages for sensitivity analysis in observational studies. *Observational Studies*, 1(2):1–17.

Rosenbaum, P. R. and Rosenbaum, P. R. (2002). *Overt bias in observational studies*. Springer.

Rosenbaum, P. R. and Rubin, D. B. (1983). The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55.

Rosenbaum, P. R. and Rubin, D. B. (1985). The bias due to incomplete matching. *Biometrics*, pages 103–116.

Rosenzweig, M. R. and Wolpin, K. I. (2000). Natural "natural experiments" in economics. *Journal of Economic Literature*, 38(4):827–874.

Rossi, P. E. (2014). Even the rich can make themselves poor: A critical examination of iv methods in marketing applications. *Marketing Science*, 33(5):655–672.

Roth, J. (2022). Pretest with caution: Event-study estimates after testing for parallel trends. *American Economic Review*, 4(3):305–322.

Roth, J. and Sant'Anna, P. H. (2023). When is parallel trends sensitive to functional form? *Econometrica*, 91(2):737–747.

Rubin, D. B. (1973). The use of matched sampling and regression adjustment to remove bias in observational studies. *Biometrics*, pages 185–203.

Rubin, D. B. (1996). Multiple imputation after 18+ years. *Journal of the American statistical Association*, 91(434):473–489.

Rubin, D. B. (2001). Using propensity scores to help design observational studies: application to the tobacco litigation. *Health Services and Outcomes Research Methodology*, 2:169–188.

Rubin, D. B. and Thomas, N. (1996). Matching using estimated propensity scores: relating theory to practice. *Biometrics*, pages 249–264.

Rubin, D. B. and Thomas, N. (2000). Combining propensity score matching with additional adjustments for prognostic covariates. *Journal of the American Statistical Association*, 95(450):573–585.

Ryan, A. M., Kontopantelis, E., Linden, A., and Burgess Jr, J. F. (2019). Now trending: Coping with non-parallel trends in difference-in-differences analysis. *Statistical methods in medical research*, 28(12):3697–3711.

Sant'Anna, P. H. and Zhao, J. (2020). Doubly robust difference-in-differences estimators. *Journal of Econometrics*, 219(1):101–122.

Sargan, J. D. (1958). The estimation of economic relationships using instrumental variables. *Econometrica: Journal of the econometric society*, pages 393–415.

Satterthwaite, F. E. (1946). An approximate distribution of estimates of variance components. *Biometrics bulletin*, 2(6):110–114.

Schabenberger, O. and Pierce, F. J. (2001). *Contemporary statistical models for the plant and soil sciences*. CRC press.

Scharfstein, D. O., Rotnitzky, A., and Robins, J. M. (1999). Adjusting for nonignorable drop-out using semiparametric nonresponse models. *Journal of the American Statistical Association*, 94(448):1096–1120.

Scheve, K. and Stasavage, D. (2012). Democracy, war, and wealth: lessons from two centuries of inheritance taxation. *American Political Science Review*, 106(1):81–102.

Schipper, K. and Smith, A. (1983). Effects of recontracting on shareholder wealth: The case of voluntary spin-offs. *Journal of financial Economics*, 12(4):437–467.

Schipper, K. and Thompson, R. (1983). Evidence on the capitalized value of merger activity for acquiring firms. *Journal of financial Economics*, 11(1-4):85–119.

Schisterman, E. F., Moysich, K. B., England, L. J., and Rao, M. (2003). Estimation of the correlation coefficient using the bayesian approach and its applications for epidemiologic research. *BMC medical research methodology*, 3:1–4.

Schuirmann, D. J. (1987). A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of pharmacokinetics and biopharmaceutics*, 15:657–680.

Schulze, C., Skiera, B., and Wiesel, T. (2012). Linking customer and financial metrics to shareholder value: The leverage effect in customer-based valuation. *Journal of Marketing*, 76(2):17–32.

Shapiro, S. S. and Francia, R. (1972). An approximate analysis of variance test for normality. *Journal of the American statistical Association*, 67(337):215–216.

Shapiro, S. S. and Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611.

Sherman, L. W. and Berk, R. A. (1984). *The Minneapolis domestic violence experiment*, volume 1. Police Foundation Washington, DC.

Shin, J., Sudhir, K., and Yoon, D.-H. (2012). When to "fire" customers: Customer cost-based pricing. *Management Science*, 58(5):932–947.

Shrout, P. E. and Bolger, N. (2002). Mediation in experimental and nonexperimental studies: new procedures and recommendations. *Psychological methods*, 7(4):422.

Šidák, Z. (1967). Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American Statistical Association*, 62(318):626–633.

Silva, J. S. and Tenreyro, S. (2006). The log of gravity. *The Review of Economics and statistics*, 88(4):641–658.

Silva, J. S. and Tenreyro, S. (2011). Further simulation evidence on the performance of the poisson pseudo-maximum likelihood estimator. *Economics Letters*, 112(2):220–222.

Simonsohn, U., Nelson, L. D., and Simmons, J. P. (2014a). P-curve: a key to the file-drawer. *Journal of experimental psychology: General*, 143(2):534.

Simonsohn, U., Nelson, L. D., and Simmons, J. P. (2014b). p-curve and effect size: Correcting for publication bias using only significant results. *Perspectives on Psychological Science*, 9(6):666–681.

Simonsohn, U., Simmons, J. P., and Nelson, L. D. (2015). Better p-curves: Making p-curve analysis more robust to errors, fraud, and ambitious p-hacking, a reply to ulrich and miller (2015).

Simonsohn, U., Simmons, J. P., and Nelson, L. D. (2020). Specification curve analysis. *Nature Human Behaviour*, 4(11):1208–1214.

Singh, A., Hosanagar, K., and Gandhi, A. (2020). Machine learning instrument variables for causal inference. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 835–836.

Skiera, B., Bayer, E., and Schöler, L. (2017). What should be the dependent variable in marketing-related event studies? *International Journal of Research in Marketing*, 34(3):641–659.

Smith, E. R. (1982). Beliefs, attributions, and evaluations: Nonhierarchical models of mediation in social cognition. *Journal of Personality and Social Psychology*, 43(2):248.

Smith, J. A. and Todd, P. E. (2005). Does matching overcome lalonde's critique of nonexperimental estimators? *Journal of econometrics*, 125(1-2):305–353.

Snedecor, G. W. and Cochran, W. G. (1989). Statistical methods.

Sobel, M. E. (1982). Asymptotic confidence intervals for indirect effects in structural equation models. *Sociological methodology*, 13:290–312.

Somaini, P. and Wolak, F. A. (2016). An algorithm to estimate the two-way fixed effects model. *Journal of Econometric Methods*, 5(1):143–152.

Somaini, P. and Wolak, F. A. (2021). Twfem: Stata module to efficiently estimate a two-way fixed effects model based on somaini and wolak (2015).

Sood, A. and Tellis, G. J. (2009). Do innovations really pay off? total stock market returns to innovation. *Marketing Science*, 28(3):442–456.

Sorescu, A., Warren, N. L., and Ertekin, L. (2017). Event study methodology in the marketing literature: an overview. *Journal of the Academy of Marketing Science*, 45:186–207.

Sorescu, A. B., Chandy, R. K., and Prabhu, J. C. (2003). Sources and financial consequences of radical innovation: Insights from pharmaceuticals. *Journal of marketing*, 67(4):82–102.

Sorescu, A. B., Chandy, R. K., and Prabhu, J. C. (2007). Why some acquisitions do better than others: product capital as a driver of long-term stock returns. *Journal of Marketing Research*, 44(1):57–72.

Spiller, S. A., Fitzsimons, G. J., Lynch Jr, J. G., and McClelland, G. H. (2013). Spotlights, floodlights, and the magic number zero: Simple effects tests in moderated regression. *Journal of marketing research*, 50(2):277–288.

St Laurent, R. T. and Cook, R. D. (1992). Leverage and superleverage in nonlinear regression. *Journal of the American Statistical Association*, 87(420):985–990.

Staiger, D. and Stock, J. H. (1997). Instrumental variables regression with weak instruments. *Econometrica*, 65(3):557–586.

Steinmetz, H. and Block, J. (2022). Meta-analytic structural equation modeling (masem): new tricks of the trade. *Management Review Quarterly*, 72(3):605–626.

Stephens, M. A. (1974). Edf statistics for goodness of fit and some comparisons. *Journal of the American statistical Association*, 69(347):730–737.

Stevenson, B. and Wolfers, J. (2006). Bargaining in the shadow of the law: Divorce laws and family distress. *The Quarterly Journal of Economics*, 121(1):267–288.

Stock, J. and Yogo, M. (2005a). Asymptotic distributions of instrumental variables statistics with many instruments. *Identification and inference for econometric models: Essays in honor of Thomas Rothenberg*, 6:109–120.

Stock, J. H. and Wright, J. H. (2000). Gmm with weak identification. *Econometrica*, 68(5):1055–1096.

Stock, J. H., Wright, J. H., and Yogo, M. (2002). A survey of weak instruments and weak identification in generalized method of moments. *Journal of Business & Economic Statistics*, 20(4):518–529.

Stock, J. H. and Yogo, M. (2002). Testing for weak instruments in linear iv regression.

Stock, J. H. and Yogo, M. (2005b). Testing for weak instruments in linear iv regression. *Identification and Inference for Econometric Models: Essays in Honor of Thomas Rothenberg*, page 80.

Strong, N. (1992). Modelling abnormal returns: A review article. *Journal of Business Finance & Accounting*, 19(4):533–553.

Stuart, E. A. (2010). Matching methods for causal inference: A review and a look forward. *Statistical science: a review journal of the Institute of Mathematical Statistics*, 25(1):1.

Student (1908). The probable error of a mean. *Biometrika*, pages 1–25.

Sudhir, K. and Talukdar, D. (2015). The "peter pan syndrome" in emerging markets: The productivity-transparency trade-off in it adoption. *Marketing Science*, 34(4):500–521.

Sun, B., Liu, L., Miao, W., Wirth, K., Robins, J., and Tchetgen, E. J. T. (2018). Semiparametric estimation with data missing not at random using an instrumental variable. *Statistica Sinica*, 28(4):1965.

Sun, C. and Liao, X. (2011). Effects of litigation under the endangered species act on forest firm values. *Journal of forest economics*, 17(4):388–398.

Sun, L. and Abraham, S. (2021). Estimating dynamic treatment effects in event studies with heterogeneous treatment effects. *Journal of Econometrics*, 225(2):175–199.

Sun, L., Ben-Michael, E., and Feller, A. (2023). Using multiple outcomes to improve the synthetic control method. *arXiv preprint arXiv:2311.16260*.

Sun, L. and Shapiro, J. M. (2022). A linear panel model with heterogeneous coefficients and variation in exposure. *Journal of Economic Perspectives*, 36(4):193–204.

Sun, M. and Zhu, F. (2013). Ad revenue and content commercialization: Evidence from blogs. *Management Science*, 59(10):2314–2331.

Swamy, P. and Arora, S. S. (1972). The exact finite sample properties of the estimators of coefficients in the error components regression models. *Econometrica: journal of the Econometric Society*, pages 261–275.

Tchetgen Tchetgen, E. (2014). A note on the control function approach with an instrumental variable and a binary outcome. *Epidemiologic methods*, 3(1):107–112.

Tchetgen Tchetgen, E. J. and Wirth, K. E. (2017). A general instrumental variable framework for regression analysis with outcome missing not at random. *Biometrics*, 73(4):1123–1131.

Tellis, G. J. and Johnson, J. (2007). The value of quality. *Marketing Science*, 26(6):758–773.

Thistlethwaite, D. L. and Campbell, D. T. (1960). Regression-discontinuity analysis: An alternative to the ex post facto experiment. *Journal of Educational psychology*, 51(6):309.

Thoemmes, F., Liao, W., and Jin, Z. (2017). The analysis of the regression-discontinuity design in r. *Journal of Educational and Behavioral Statistics*, 42(3):341–360.

Tibbe, T. D. and Montoya, A. K. (2022). Correcting the bias correction for the bootstrap confidence interval in mediation analysis. *Frontiers in psychology*, 13:810258.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288.

Tipton, M. M., Bharadwaj, S. G., and Robertson, D. C. (2009). Regulatory exposure of deceptive marketing and its impact on firm value. *Journal of Marketing*, 73(6):227–243.

Tirunillai, S. and Tellis, G. J. (2012). Does chatter really matter? dynamics of user-generated content and stock performance. *Marketing Science*, 31(2):198–215.

Tirunillai, S. and Tellis, G. J. (2017). Does offline tv advertising affect online chatter? quasi-experimental analysis using synthetic control. *Marketing Science*, 36(6):862–878.

Tukey, J. W. (1993). Tightening the clinical trial. *Controlled clinical trials*, 14(4):266–285.

Utts, J. M. (1982). The rainbow test for lack of fit in regression. *Communications in Statistics-Theory and Methods*, 11(24):2801–2815.

Vach, W. and Vach, W. (1994). Missing values and subsampling. *Logistic Regression with Missing Values in the Covariates*, pages 98–102.

VanderWeele, T. (2015). *Explanation in causal inference: methods for mediation and interaction*. Oxford University Press.

VanderWeele, T. J. (2019). Principles of confounder selection. *European journal of epidemiology*, 34:211–219.

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

Von Hippel, P. T. (2009). 8. how to impute interactions, squares, and other transformed variables. *Sociological methodology*, 39(1):265–291.

Vuong, Q. H. (1989). Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica: journal of the Econometric Society*, pages 307–333.

Wallace, T. D. and Hussain, A. (1969). The use of error components models in combining cross section with time series data. *Econometrica: Journal of the Econometric Society*, pages 55–72.

Wang, Y. and Bellemare, M. F. (2019). Lagged variables as instruments. Technical report, Working Paper, Department of Applied Economics, University of Minnesota.

Wang, Y., Lewis, M., and Schweidel, D. A. (2018). A border strategy analysis of ad source and message tone in senatorial campaigns. *Marketing Science*, 37(3):333–355.

Wang, Y., Wu, C., and Zhu, T. (2019). Mobile hailing technology and taxi driving behaviors. *Marketing Science*, 38(5):734–755.

Warren, N. L. and Sorescu, A. (2017). When 1+ 1> 2: How investors react to new product releases announced concurrently with other corporate news. *Journal of Marketing*, 81(2):64–82.

Watson, G. S. (1964). Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 359–372.

Webb, A. R., Copsey, K. D., and Cawley, G. (2011). *Statistical pattern recognition*, volume 2. Wiley Online Library.

Westfall, J. and Yarkoni, T. (2016). Statistically controlling for confounding constructs is harder than you think. *PloS one*, 11(3):e0152719.

White, H. (1980). A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica: journal of the Econometric Society*, pages 817–838.

Wiles, M. A. and Danielova, A. (2009). The worth of product placement in successful films: An event study analysis. *Journal of marketing*, 73(4):44–63.

Wiles, M. A., Jain, S. P., Mishra, S., and Lindsey, C. (2010). Stock market response to regulatory reports of deceptive advertising: The moderating effect of omission bias and firm reputation. *Marketing Science*, 29(5):828–845.

Wiles, M. A., Morgan, N. A., and Rego, L. L. (2012). The effect of brand acquisition and disposal on stock returns. *Journal of Marketing*, 76(1):38–58.

Wing, C., Yozwiak, M., Hollingsworth, A., Freedman, S., and Simon, K. (2024). Designing difference-in-difference studies with staggered treatment adoption: Key concepts and practical guidelines. *Annual Review of Public Health*, 45.

Wolfers, J. (2003). Is business cycle volatility costly? evidence from surveys of subjective well-being. *International finance*, 6(1):1–26.

Wolfolds, S. E. and Siegel, J. (2019). Misaccounting for endogeneity: The peril of relying on the heckman two-step method without a valid instrument. *Strategic Management Journal*, 40(3):432–462.

Wooldridge, J. M. (1999). Quasi-likelihood methods for count data. *Handbook of applied econometrics volume 2: Microeconomics*, pages 321–368.

Wooldridge, J. M. (2022). Simple approaches to nonlinear difference-in-differences with panel data. *Available at SSRN 4183726*.

Wooldridge, J. M. (2023). Simple approaches to nonlinear difference-in-differences with panel data. *The Econometrics Journal*, 26(3):C31–C66.

Wu, Q., Luo, X., Slotegraaf, R. J., and Aspara, J. (2015). Sleeping with competitors: the impact of npd phases on stock market reactions to horizontal collaboration. *Journal of the Academy of Marketing Science*, 43:490–511.

Xiong, G. and Bharadwaj, S. (2013). Asymmetric roles of advertising and marketing capability in financial returns to news: Turning bad into good and good into great. *Journal of Marketing Research*, 50(6):706–724.

Xu, Y. (2017). Generalized synthetic control method: Causal inference with interactive fixed effects models. *Political Analysis*, 25(1):57–76.

Yang, S., Imbens, G. W., Cui, Z., Faries, D. E., and Kadziola, Z. (2016). Propensity score matching and subclassification in observational studies with multilevel treatments. *Biometrics*, 72(4):1055–1065.

Yao, L., Li, S., Li, Y., Huai, M., Gao, J., and Zhang, A. (2018). Representation learning for treatment effect estimation from observational data. *Advances in neural information processing systems*, 31.

Yule, G. U. (1899). An investigation into the causes of changes in pauperism in england, chiefly during the last two intercensal decades (part i.). *Journal of the Royal Statistical Society*, 62(2):249–295.

Zhang, J. L. and Rubin, D. B. (2003). Estimation of causal effects via principal stratification when some outcomes are truncated by "death". *Journal of Educational and Behavioral Statistics*, 28(4):353–368.

Zhang, S., Lee, D., Singh, P. V., and Srinivasan, K. (2022). What makes a good image? airbnb demand analytics leveraging interpretable image features. *Management Science*, 68(8):5644–5666.

Zhao, Q.-Y., Luo, J.-C., Su, Y., Zhang, Y.-J., Tu, G.-W., and Luo, Z. (2021). Propensity score matching with r: conventional methods and new features. *Annals of translational medicine*, 9(9).

Zou, G. Y. (2008). On the estimation of additive interaction by use of the four-by-two table and beyond. *American journal of epidemiology*, 168(2):212–224.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320.