Fantasy Football Analytics: Statistics, Prediction, and Empiricism Using R

Version 0.0.1

Isaac T. Petersen

To our daughter, Maisie.

Table of contents

Preface **xix**

How to Contribute . . . xix

Open Access . . . xix

License . . . xx

Citation . . . xx

About the Author . . . xxi

Accessibility . . . xxii

Acknowledgments . . . xxii

1 Introduction **1**

1.1 About this Book . . . 1

1.2 What is Fantasy Football? . . . 1

1.3 Why Focus on Fantasy Football? . . . 2

1.4 Why R? . . . 2

1.5 Educational Value . . . 3

1.6 Learning Objectives . . . 4

1.7 Disclosures . . . 5

1.8 Disclaimer . . . 5

2 Intro to Football and Fantasy **7**

2.1 Football . . . 7

2.1.1 The Objective . . . 7

2.1.2 The Roster . . . 7

2.1.3 The Field . . . 10

2.1.4 The Gameplay . . . 12

2.1.5 The Scoring 13
2.1.6 Glossary of Terms 13
2.2 Fantasy Football 15
2.2.1 Overview of Fantasy Football 15
2.2.2 The Fantasy League 16
2.2.3 The Roster of a Fantasy Team 16
2.2.4 Scoring 17

3 Getting Started with R for Data Analysis 21

3.1 Learning R 21
3.2 Getting Help with R 22
3.3 Initial Setup 22
3.4 Installing Packages 24
3.5 Load Packages 24
3.6 Using Functions and Arguments 25
3.7 Create a Vector 28
3.8 Create a Data Frame 28
3.9 Create a List 29
3.10 Load a Data Frame 29
3.11 Variable Names 30
3.12 Logical Operators 30
3.12.1 Is Equal To: `==` 30
3.12.2 Is Not Equal To: `!=` 31
3.12.3 Is Greater Than: `>` 31
3.12.4 Is Less Than: `<` 31
3.12.5 Is Greater Than or Equal To: `>=` 31
3.12.6 Is Less Than or Equal To: `<=` 31
3.12.7 Is In a Value of Another Vector: `%in%` 32
3.12.8 Is Not In a Value of Another Vector: `!(%in%)` 32
3.12.9 Is Missing: `is.na()` 32
3.12.10 Is Not Missing: `!is.na()` 32

3.12.11 And: & . . . 32
3.12.12 Or: | . . . 33
3.13 Piping . . . 33
3.14 Subset . . . 35
3.14.1 One Variable . . . 35
3.14.2 Particular Rows of One Variable . . . 35
3.14.3 Particular Columns (Variables) . . . 36
3.14.4 Particular Rows . . . 41
3.14.5 Particular Rows and Columns . . . 42
3.15 View Data . . . 43
3.15.1 All Data . . . 43
3.15.2 First 6 Rows/Elements . . . 43
3.16 Data Characteristics . . . 44
3.16.1 Data Structure . . . 44
3.16.2 Data Dimensions . . . 45
3.16.3 Number of Elements . . . 46
3.16.4 Number of Missing Elements . . . 46
3.16.5 Number of Non-Missing Elements . . . 46
3.17 Create New Variables . . . 46
3.18 Recode Variables . . . 47
3.19 Rename Variables . . . 48
3.20 Convert the Types of Variables . . . 48
3.21 Merging/Joins . . . 51
3.21.1 Overview . . . 51
3.21.2 Data Before Merging . . . 51
3.21.3 Types of Joins . . . 53
3.22 Transform Data from Long to Wide . . . 61
3.23 Transform Data from Wide to Long . . . 81
3.24 Loops . . . 82

4 Download and Process NFL Football Data 85

4.1 Load Packages 85

4.2 Data Dictionaries of NFL Data 85

4.3 Types of NFL Data 86

4.3.1 Players 86

4.3.2 Teams 87

4.3.3 Fantasy Player IDs 88

4.3.4 Player Info 90

4.3.5 Rosters 90

4.3.6 Game Schedules 94

4.3.7 Standings 95

4.3.8 The Combine 95

4.3.9 Draft Picks 100

4.3.10 Draft Values 101

4.3.11 Depth Charts 103

4.3.12 Trades 104

4.3.13 Play-By-Play Data 105

4.3.14 4th Down Data 107

4.3.15 Participation 108

4.3.16 Historical Weekly Actual Player Statistics 109

4.3.17 Injuries 111

4.3.18 Snap Counts 112

4.3.19 ESPN QBR 113

4.3.20 NFL Next Gen Stats 116

4.3.21 Advanced Stats from Pro Football Reference 117

4.3.22 Player Contracts 131

4.3.23 FTN Charting Data 132

4.3.24 FantasyPros Rankings 133

4.3.25 Expected Fantasy Points 135

4.3.26 Fantasy Football Projections 140

4.4 Calculations 156

4.4.1 Historical Actual Player Statistics 156
4.4.2 Historical Actual Fantasy Points 167
4.4.3 Player Age and Experience 167

5 Data Visualization 181

5.1 Getting Started 181
5.1.1 Load Packages 181
5.1.2 Load Data 181
5.2 Overview 181
5.2.1 Principles of Graphic Design 181
5.2.2 Principles of Data Visualization 182
5.2.3 Creating Data Visualizations in R 184
5.3 Univariate Distribution 184
5.3.1 Histogram 184
5.3.2 Density Plot 185
5.3.3 Histogram with Overlaid Density and Rug Plot 186
5.3.4 Box-and-Whisker Plot 187
5.3.5 Violin Plot 188
5.4 Scatterplot 189
5.4.1 Base Layer 190
5.4.2 Add Points 190
5.4.3 Best-Fit Line 191
5.4.4 Modify Axes 195
5.4.5 Plot Labels 196
5.4.6 Theme 197
5.4.7 Interactive 200
5.5 Line Chart 200
5.5.1 With Highlighting 202
5.6 Bar Plot 203
5.6.1 With Error Bars 204
5.6.2 Modified Color Scheme 206

5.7 Examples . . . 207
5.7.1 Players . . . 207
5.7.2 Teams . . . 210
5.8 Conclusion . . . 212

6 Player Evaluation 213
6.1 Getting Started . . . 213
6.1.1 Load Packages . . . 213
6.2 Overview . . . 213
6.3 Athletic Profile . . . 214
6.4 Skill . . . 215
6.5 Historical Performance . . . 215
6.5.1 Overview . . . 215
6.5.2 Efficiency . . . 216
6.5.3 Consistency . . . 217
6.6 Health . . . 218
6.7 Availability . . . 219
6.8 Age and Career Stage . . . 219
6.9 Situational Factors . . . 219
6.10 Matchups . . . 220
6.11 Cognitive and Motivational Factors . . . 221
6.12 Fantasy Value . . . 221
6.12.1 Sources From Which to Evaluate Fantasy Value . . . 221
6.12.2 Indices to Evaluate Fantasy Value . . . 223
6.13 Signs Versus Samples . . . 226
6.14 Putting it Altogether . . . 228

7 The Fantasy Draft 229
7.1 Getting Started . . . 229
7.1.1 Load Packages . . . 229
7.2 Types of Fantasy Drafts . . . 229

7.2.1 Snake Draft 229
7.2.2 Auction Draft 229
7.2.3 Comparison 230
7.3 Draft Strategy 230
7.3.1 Overview 230
7.3.2 Snake Draft 233
7.3.3 Auction Draft 233

8 Research Methods 235
8.1 Getting Started 235
8.1.1 Load Packages 235
8.2 Sample vs Population 235
8.3 Research Designs 236
8.3.1 Experiment 236
8.3.2 Correlational/Observational Study 237
8.3.3 Case Study 239
8.3.4 Other Features of the Research Design 240
8.4 Research Design Validity 242
8.4.1 Internal Validity 243
8.4.2 External Validity 243
8.4.3 Tradeoffs Between Internal and External Validity 243
8.4.4 Conclusion Validity 244
8.5 Mediation vs Moderation 244
8.5.1 Mediation 245
8.5.2 Moderation (i.e., Interaction) 247
8.6 Levels of Measurement 250
8.6.1 Nominal 250
8.6.2 Ordinal 250
8.6.3 Interval 251
8.6.4 Ratio 251
8.7 Psychometrics 252

8.7.1 Measurement Reliability 252
8.7.2 Measurement Validity 254
8.7.3 Reliability vs Validity 257
8.8 Conclusion 257

9 Basic Statistics 259
9.1 Getting Started 259
9.1.1 Load Packages 259
9.2 Descriptive Statistics 259
9.2.1 Center 259
9.2.2 Spread 261
9.2.3 Shape 262
9.2.4 Combination 262
9.3 Scores and Scales 263
9.3.1 Raw Scores 263
9.3.2 z Scores 263
9.4 Inferential Statistics 267
9.4.1 Null Hypothesis Significance Testing 268
9.4.2 Practical Significance 281
9.5 Statistical Decision Tree 284
9.6 Statistical Tests 286
9.6.1 t-Test 286
9.6.2 Analysis of Variance 288
9.6.3 Correlation 289
9.6.4 (Multiple) Regression 289
9.6.5 Chi-Square Test 289
9.6.6 Formulating Statistical Tests in Terms of Partitioned Variance 290
9.6.7 Critical Value 290
9.6.8 Statistical Power 297
9.7 Conclusion 321

10 Correlation Analysis **323**
10.1 Getting Started . . . 323
10.1.1 Load Packages . . . 323
10.2 Overview of Correlation . . . 323
10.3 The Correlation Coefficient (r) . . . 323
10.4 Examples . . . 336
10.4.1 Covariance . . . 336
10.4.2 Pearson Correlation . . . 336
10.4.3 Spearman Correlation . . . 336
10.4.4 Nonlinear Correlation . . . 336
10.4.5 Correlation Matrix . . . 336
10.4.6 Correlogram . . . 336
10.5 Impact of Outliers . . . 336
10.6 Correlation Does Not Imply Causation . . . 336
10.7 Conclusion . . . 336
10.8 Session Info . . . 337

11 Multiple Regression **339**
11.1 Getting Started . . . 339
11.1.1 Load Packages . . . 339
11.2 Overview of Multiple Regression . . . 339
11.3 Components . . . 340
11.4 Assumptions of Multiple Regression . . . 341
11.5 Coefficient of Determination (R^2) . . . 341
11.6 Overfitting . . . 344
11.7 Covariates . . . 346
11.8 Multicollinearity . . . 347
11.9 Impact of Oultiers . . . 349
11.10 Moderated Multiple Regression . . . 349
11.11 Mediation . . . 350
11.12 Bayesian Multiple Regression . . . 350
11.13 Conclusion . . . 350

12 Mixed Models **351**

12.1 Getting Started 351

12.1.1 Load Packages 351

12.1.2 Specify Package Options 351

12.1.3 Load Data 352

12.2 Overview of Mixed Models 352

12.2.1 Ecological Fallacy 353

12.2.2 Simpson's Paradox 353

12.3 Fantasy Points Per Season by Position, Age, and Experience . . 355

12.3.1 Scatterplots of Fantasy Points by Age and Position . . . 356

12.3.2 Plots of Raw Trajectories of Fantasy Points By Age and Player 368

12.3.3 Linear Regression Models 379

12.3.4 Mixed Models 388

12.3.5 Bayesian Mixed Models 451

12.3.6 Plots of Model-Implied Fantasy Points by Position and Age 464

12.3.7 Plots of Individuals' Model-Implied Fantasy Points by Age 469

12.3.8 Summary of Findings 485

12.4 Conclusion 486

13 Causal Inference **487**

13.1 Getting Started 487

13.1.1 Load Packages 487

13.2 Correlation Does Not Imply Causation 487

13.3 Criteria for Causality 487

13.4 Approaches for Causal Inference 490

13.4.1 Experimental Designs 490

13.4.2 Quasi-Experimental Designs 490

13.5 Causal Diagrams 498

13.5.1 Overview 498

13.5.2 Confounding 506
13.5.3 Mediation 508
13.5.4 Ancestors and Descendants 512
13.5.5 Collider Bias 514
13.5.6 Selection Bias 521
13.6 Conclusion 521

14 Heuristics and Cognitive Biases in Prediction 523
14.1 Getting Started 523
14.1.1 Load Packages 523
14.2 Overview 523
14.3 Examples of Heuristics 525
14.3.1 Availability Heuristic 525
14.3.2 Representativeness Heuristic 525
14.3.3 Anchoring and Adjustment Heuristic 526
14.3.4 Affect Heuristic 526
14.3.5 Performance Heuristic 526
14.3.6 What You See Is All There Is (WYSIATI) Heuristic . . 527
14.4 Examples of Cognitive Biases 527
14.4.1 Overconfidence Bias 528
14.4.2 Optimism Bias 530
14.4.3 Confirmation Bias 530
14.4.4 In-Group Bias 530
14.4.5 Hindsight Bias 531
14.4.6 Outcome Bias 531
14.4.7 Self-Serving Bias 531
14.4.8 Omission Bias 532
14.4.9 Loss Aversion Bias 533
14.4.10 Primacy Effect Bias 533
14.4.11 Recency Effect Bias 533
14.4.12 Framing Effect Bias 533

14.4.13 Endowment Effect Bias 534
14.4.14 Bandwagon Effect Bias 534
14.4.15 Dunning–Kruger Effect Bias 534
14.5 Examples of Fallacies 535
14.5.1 Base Rate Fallacy 536
14.5.2 Regression Fallacy 537
14.5.3 Hot Hand Fallacy 538
14.5.4 Sunk Cost Fallacy 539
14.5.5 Gambler's Fallacy 539
14.5.6 Anecdotal Fallacy 542
14.5.7 Conditional Probability Fallacy 542
14.5.8 Ecological Fallacy 542
14.6 Conclusion 542

15 Judgment Versus Actuarial Approaches to Prediction 545
15.1 Getting Started 545
15.1.1 Load Packages 545
15.2 Approaches to Prediction 545
15.2.1 Human Judgment 545
15.2.2 Actuarial/Statistical Method 546
15.2.3 Combining Human Judgment and Statistical Algorithms 547
15.3 Errors in Human Judgment 548
15.4 Humans Versus Computers 549
15.4.1 Advantages of Computers 549
15.4.2 Advantages of Humans 549
15.4.3 Comparison of Evidence 551
15.5 Why Judgment is More Widely Used Than Statistical Formulas 551
15.6 Steps to Conduct Actuarial Approaches 552
15.7 Challenges of Data-Driven Approaches and How to Address . . 553
15.7.1 Shrinkage 553

15.7.2 Cross-Validation 553
15.8 Best Actuarial Approaches to Prediction 554
15.9 Conclusion 554

16 Base Rates 555
16.1 Getting Started 555
16.1.1 Load Packages 555
16.2 Overview 555
16.3 Issues Around Probability 556
16.3.1 Types of Probabilities 556
16.3.2 Confusion of the Inverse 557
16.3.3 Bayes' Theorem 558
16.4 Base Rate of Rookie Performance 562
16.4.1 Quarterbacks 562
16.4.2 Running Backs 562
16.5 How to Account for Base Rates 562
16.5.1 Actuarial Formula 563
16.5.2 Bayesian Updating 563
16.6 Conclusion 569

17 Evaluation of Prediction/Forecasting Accuracy 571
17.1 Getting Started 571
17.1.1 Load Packages 571
17.2 Overview 571
17.3 Types of Accuracy 572
17.3.1 Discrimination 573
17.3.2 Calibration 573
17.3.3 General Accuracy 573
17.4 Prediction of Categorical Outcomes 574
17.5 Prediction of Continuous Outcomes 574
17.6 Threshold-Dependent Accuracy Indices 574

17.6.1 Decision Outcomes 574
17.6.2 Percent Accuracy 576
17.6.3 Percent Accuracy by Chance 576
17.6.4 Predicting from the Base Rate 577
17.6.5 Different Kinds of Errors Have Different Costs 578
17.6.6 Sensitivity, Specificity, PPV, and NPV 579
17.6.7 Signal Detection Theory 589
17.6.8 Accuracy Indices 595
17.7 Threshold-Independent Accuracy Indices 606
17.7.1 General Prediction Accuracy 606
17.7.2 Discrimination 612
17.7.3 Calibration 613
17.8 Integrating the Accuracy Indices 620
17.9 Theory Versus Empiricism 621
17.10 Test Bias 624
17.11 Ways to Improve Prediction Accuracy 624
17.12 Conclusion 626

18 Mythbusters: Putting Fantasy Football Beliefs/Anecdotes to the Test 627

18.1 Getting Started 627
18.1.1 Load Packages 627
18.1.2 Specify Package Options 627
18.1.3 Load Data 627
18.2 Do Players Perform Better in their Contract Year? 628
18.2.1 QB 632
18.2.2 RB 642
18.2.3 WR/TE 647
18.2.4 QB/RB/WR/TE 652
18.3 Conclusion 652

19 Modern Portfolio Theory **653**

19.1 Getting Started 653

19.1.1 Load Packages 653

19.2 Overview 653

19.3 Fantasy Football is Like Stock Picking 653

19.3.1 The Wisdom of the Crowd (or Market) 653

19.3.2 Diversification 654

19.4 The Efficient Frontier of a Stock Portfolio 656

19.4.1 Download Historical Stock Prices 657

19.4.2 Calculate Stock Returns 657

19.4.3 Create Portfolio 657

19.4.4 Determine the Efficient Frontier 658

19.4.5 Identify the Optimal Weights 660

19.5 The Efficient Frontier of a Fantasy Team 664

19.5.1 Based on Variability Across Projection Sources 664

19.5.2 Based on Historical Game-to-Game Variability 664

19.6 Conclusion 664

20 Cluster Analysis **665**

20.1 Getting Started 665

20.1.1 Load Packages 665

20.1.2 Load Data 665

20.1.3 Overview 665

20.1.4 Tiers of Prior Season Fantasy Points 666

20.1.5 Types of Wide Receivers 690

20.2 Conclusion 702

21 Time Series Analysis **703**

21.1 Getting Started 703

21.1.1 Load Packages 703

21.1.2 Load Data 703

21.2 Overview of Time Series Analysis 703
21.3 Autoregressive Integrated Moving Average (ARIMA) Models . 704
21.3.1 Create the Time Series Objects 704
21.3.2 Plot the Time Series 706
21.3.3 Rolling Mean/Median 708
21.3.4 Autocorrelation 709
21.3.5 Fit an Autoregressive Integrated Moving Average Model 710
21.3.6 Generate the Model Forecasts 716
21.3.7 Plot the Model Forecasts 717
21.4 Bayesian Mixed Models 719
21.4.1 Prepare New Data Object 719
21.4.2 Generate Predictions 721
21.4.3 Table of Next Season Predictions 721
21.4.4 Plot of Individuals' Model-Implied Predictions 723
21.5 Conclusion 732

22 Decision Making in the Context of Uncertainty 733
22.1 Getting Started 733
22.1.1 Load Packages 733
22.2 Overview of Decision Making in Uncertainty 733
22.3 Wisdom of the Crowd 733
22.4 Accuracy of Fantasy Football Crowd Projections 735
22.5 How Well Do People Incorporate Advice From Others? 739
22.6 Suggestions 739
22.7 Conclusion 740

References 741

Preface

This is a book in progress—it is incomplete. I will continue to add to and update it as I am able.

How to Contribute

This is an open-access textbook. My goal is to share data analysis strategies for free! Anyone is welcome to contribute to the project. If you would like to contribute, please consider one of the following:

- open an issue[1] or create a pull request[2] on the book's GitHub repository[3].
- buy me a coffee[4]—Support me in developing this (free!) resource for fantasy football analytics… Even a cup of coffee helps me stay awake!

The GitHub repository for the book is located here: https://github.com/isaactpetersen/Fantasy-Football-Analytics-Textbook. If you have data or analysis examples that are you willing to share and include in the book, feel free to contact me.

Open Access

This is an open-access book. This means that it is freely available for anyone to access.

[1] https://github.com/isaactpetersen/Fantasy-Football-Analytics-Textbook/issues
[2] https://github.com/isaactpetersen/Fantasy-Football-Analytics-Textbook/pulls
[3] https://github.com/isaactpetersen/Fantasy-Football-Analytics-Textbook
[4] https://www.buymeacoffee.com/isaactpetersen

License

Figure 1 Creative Commons License

The online version of this book is licensed under the Creative Commons Attribution License[5]. In short, you can use my work as long as you cite it.

Citation

The APA-style citation for the book is:

Petersen, I. T. (2024). *Fantasy football analytics: Statistics, prediction, and empiricism Using R*. Version 0.0.1. University of Iowa Libraries. https://github.com/isaactpetersen/Fantasy-Football-Analytics-Textbook. [INSERT DOI LINK]

The BibTeX citation for the book is:

```
@book{petersenFantasyFootballAnalytics,
  title = {Fantasy football analytics: Statistics, prediction, and empiricism using {R}},
  author = {Petersen, Isaac T.},
  year = {2024},
  publisher = {{University of Iowa Libraries}},
  note = {Version 0.0.1},
  doi = {INSERT},
  isbn = {INSERT},
  url = {https://github.com/isaactpetersen/Fantasy-Football-Analytics-Textbook}
}
```

[5] https://creativecommons.org/licenses/by/4.0/

About the Author

I am an Associate Professor in the Department of Psychological and Brain Sciences at the University of Iowa. I am a licensed psychologist with expertise in child clinical psychology. Why am I writing about fantasy football and data analysis? Because fantasy football involves the intersection of two things I love: sports and statistics.

Through my training, I have learned the value of statistics for answering important questions that I find interesting. In graduate training, I came to the realization that statistics are relevant not only for psychology and science, but also for domains that I enjoy as hobbies, including sports and fantasy sports. I have played in a longstanding fantasy football league for over 20 years (since my junior year of high school) with old friends from high school. I wanted to apply what I was learning about statistics to help others improve their performance in fantasy football and to help people—including those who might not otherwise be interested—to learn statistics. So I began blogging online about the value of applying statistics to improve decision making in fantasy football. Apparently, many people were interested in learning statistics when they could apply them to a domain that they find interesting like fantasy football. My blog eventually became FantasyFootballAnalytics.net[6], a website that uses advanced statistics to help people win their fantasy football leagues.

In terms of my `R` and statistics background, I have published many peer-reviewed publications[7] that employ advanced statistical methods, have published a book on psychological assessment[8] (Petersen, 2024b, 2024c) that includes applied examples in `R`, and have published the `petersenlab` `R` package[9] (Petersen, 2024a) on the Comprehensive R Archive Network (CRAN). Several sections in this book come from Petersen (2024c). I am also a co-author of the `ffanalytics` `R` package[10] (Andersen et al., 2024) that provides free utilities for downloading fantasy football projections and additional fantasy-relevant data, and for calculating projected points given your league settings.

[6] http://fantasyfootballanalytics.net
[7] https://developmental-psychopathology.lab.uiowa.edu/publications
[8] https://www.routledge.com/9781032413068
[9] https://cran.r-project.org/web/packages/petersenlab/index.html
[10] https://github.com/FantasyFootballAnalytics/ffanalytics

Accessibility

I strive to follow principles of accessibility[11] (archived at https://perma.cc/8XJ9-Q6QJ) to make the book content accessible to people with visual impairments and physical disabilities. If there are additional ways I can make the content more accessible, please let me know.

Acknowledgments

I thank Dr. Benjamin Motz, who provided consultation and many helpful resources based on his fantasy football statistics class. I also thank key members of FantasyFootballAnalytics.net[12], including Val Pinskiy, Andrew Tungate, Dennis Andersen, and Adam Peterson, who helped develop and provide fantasy football-related resources and who helped sharpen my thinking about the topic. I also thank Professor Patrick Carroll, who taught me the value of statistics for answering important questions.

[11] https://bookdown.org/yihui/rmarkdown-cookbook/html-accessibility.html
[12] http://fantasyfootballanalytics.net

1

Introduction

1.1 About this Book

How can we use information to make predictions about uncertain events? This book is about empiricism (basing theories on observed data) and judgment, prediction, and decision making in the context of uncertainty. The book provides an introduction to modern analytical techniques used to make informed predictions, test theories, and draw conclusions from a given dataset. The book leverages the software `R` for providing applied data analysis examples.

This book was originally written for a undergraduate-level course entitled, "Fantasy Football: Predictive Analytics and Empiricism". The chapters provide an overview of topics that each could have its own class and textbook, such as causal inference[1], factor analysis[2], cluster analysis[3], principal component analysis[4], machine learning[5], cognitive biases[6], modern portfolio theory[7], data visualization[8], simulation[9], etc. The book gives readers an overview of the breadth of the approaches to prediction and empiricism. As a consequence, the book does not cover any one technique or approach in great depth.

1.2 What is Fantasy Football?

Fantasy football is an online game where participants assemble (i.e., "draft") imaginary teams composed of real-life National Football League (NFL)

[1] causal-inference.qmd
[2] factor-analysis.qmd
[3] cluster-analysis.qmd
[4] pca.qmd
[5] machine-learning.qmd
[6] cognitive-bias.qmd
[7] modern-portfolio-theory.qmd
[8] data-visualization.qmd
[9] simulation.qmd

players. In this game, participants compete against their opponents (e.g., friends/coworkers/classmates), accumulating points based on players' actual statistical performances in games. The goal is to outscore one's opponent each week to win matches and ultimately claim victory in the league.

1.3 Why Focus on Fantasy Football?

I was fortunate to have an excellent instructor who taught me the value of learning statistics to answer interesting and important questions. That is, I do not find statistics intrinsically interesting; rather, I find them interesting because of what they allow me to do. Many students find statistics intimidating in part because of how it is typically taught—with examples like dice rolls and coin flips that are (seemingly irrelevant and) boring to students. My contention is that applied examples are a more effective lens to teach many concepts in psychology and data analysis. It can be more engaging and relatable to learn statistics in the applied context of sports, a domain that is more intuitive to many. Many people play fantasy sports. This book involves applying statistics to a particular domain (football). People actually want to learn statistical principles and methods when they can apply them to interesting questions (e.g., sports). In my opinion [and supported by evidence; Motz (2013)], this is a much more effective way of engaging people and teaching statistics than in the context of abstract coin flips and dice rolls. Fantasy football relies heavily on prediction—trying to predict which players will perform best and selecting them accordingly. In this way, fantasy football provides a plethora of decision making opportunities in the face of uncertainty, and a wealth of data for analyzing these decisions. However, unlike many other applied domains in psychology, fantasy football (1) allows a person to see the accuracy of their predictions on a timely basis and (2) provides a safe environment for friendly competition. Thus, it provides a unique domain to evaluate—and improve—the accuracy of various prediction models.

1.4 Why R?

The book provides data analysis examples using the statistical analysis software, R. Why R?

- R is free! Anyone can use it.

- R is open source—it is not a black box. You can see what is going on "under the hood" and can examine the code for any function or computation you perform. You can even modify and improve these functions by changing the code, and you can create your own functions.
- R is open platform—you can use it on multiple platforms, including Windows, MacOS, and Linux.
- R has advanced statistics capabilities. It was designed for statistical analysis and has strong capabalities for data wrangling.
- R has capabilities for state-of-the-art graphics. It has advanced capabilities for creating statistical graphics.
- R is widely used—there is a large community of people who use R for data analysis that you can draw upon for help from others.
- R analyses are based on code (rather than a graphical user interface), which allows reproducibility—with the same data, code, and setup (platform, R version, package versions, etc.), you should get the same answer every time. There are strong resources available for ensuring your analyses in R are reproducible by others (Gandrud, 2020).
- Anyone (including you) can contribute R packages to the community to improve its functionality. Statistical experts from all over the world have contributed open source packages to R for specialized tasks. In the chance there is not an R package that does what you need to do, you can write a function to perform the task and can contribute it as a package to the community for others to use and improve. The number of R packages contributed to the community is growing at a rapid rate. As of this writing, over 20,000 packages have been contributed to the Comprehensive R Archive Network[10] (CRAN). And many more are stored on publicly available version control repositories like GitHub and GitLab. Chances are, if there is an analysis you need to do, an R package exists to do it.

1.5 Educational Value

Skills in statistics, statistical programming, and data analysis are highly valuable. This book includes practical and conceptual tools that build a foundation for critical thinking. The book aims to help readers evaluate theory in the light of evidence (and vice versa) and to refine decision making in the context of uncertainty. Readers will learn about the ways that psychological science (and related disciplines) poses questions, formulates hypotheses, designs studies to test those questions, and interprets the findings, collectively with the aim to answer questions, improve decision making, and solve problems.

[10] https://cran.r-project.org/web/packages/

Of course, this is not a traditional psychology textbook. However, the book incorporates important psychological concepts, such as cognitive biases in judgment and prediction, etc. In the modern world of big data, research and society need people who know how to make sense of the information around us. Psychology is in a prime position to teach applied statistics to a wide variety of students, most of whom will not have careers as psychologists. Psychology can teach the importance of statistics given humans' cognitive biases. It can also teach about how these biases can influence how people interpret statistics. This book will teach readers the applications of statistics (prediction) and research methods (empiricism) to answer questions they find interesting, while applying scientific and psychological rigor.

1.6 Learning Objectives

This book aims to help readers accomplish the following learning objectives:

- Apply empirical inference and appreciate the value it provides over speculative supposition.
- Ask educated questions when confronted with decisions in the face of uncertainty.
- Understand human decision making, including common heuristics and cognitive biases and how to mitigate them analytically.
- Engage in critical thinking about causality, including devising plausible alternative explanations for observed effects.
- Understand causal inference including confounding, causal pathways, and counterfactuals.
- Think empirically about human behavior and performance.
- Describe the strengths and weaknesses of humans versus computers in prediction scenarios.
- Apply basic skills in statistical programming using `R` to manipulate and summarize datasets and to conduct data analysis.
- Critically evaluate the strengths and limitations of different statistical models and methodologies used in predicting uncertain events, enhancing their understanding of statistical inference and model selection.
- Use various analytical techniques for predicting the outcome of uncertain events, and for uncovering latent causes of patterns in observed data.
- Interpret findings from various statistical approaches and evaluate the accuracy of predictions.
- Engage in iterative problem-solving processes, refining analytical approaches based on feedback and outcomes, and adapting strategies accordingly.

- Communicate statistical findings and analyses in both written and oral formats, demonstrating proficiency in presenting complex information to diverse audiences.
- Make sense of big data.
- Use practical analytical skills that can be applied in future research and job settings.

1.7 Disclosures

I am the Owner of Fantasy Football Analytics, LLC, which operates https://fantasyfootballanalytics.net.

1.8 Disclaimer

"*This material probably won't win you fantasy football championships. You could take what we learn and apply it to fantasy football and you might become 5 percent more likely to win. Or… Consider the broader relevance of this. You could learn data analysis and figure out ways to apply it to other systems. And you could be making a six-figure salary within the next five years.*" – Benjamin Motz, Ph.D.

2

Intro to Football and Fantasy

This chapter provides a brief primer on (American) football and fantasy football. If you are already familiar with fantasy football, feel free to skip this chapter.

2.1 Football

Football is the most widely watched sport in the United States.[1]

2.1.1 The Objective

The goal in football is for a team to score more points than their opponent. A game lasts 60 minutes, and it is separated into four 15-minute quarters. The team with the most points when the time runs out wins.

2.1.2 The Roster

2.1.2.1 Overview

Each team has 11 players on the field at a time. The particular players who are on the field will depend on the situation, but usually includes one of the three subsets of players:

1. Offense
2. Defense
3. Special Teams

[1] https://news.gallup.com/poll/610046/football-retains-dominant-position-favorite-sport.aspx (archived at https://perma.cc/X2UG-RAAK); https://www.statista.com/statistics/1430289/most-watched-sports-leagues-usa/ (archived at https://perma.cc/JNU6-S96A)

An example formation is depicted in Figure 2.1.

Figure 2.1 An Example Football Formation for the Offense and Defense. The solid line indicates the line of scrimmage. The arrow indicates the direction the offense tries to advance the ball.

2.1.2.2 Offense

The offense is on the field when the team has the ball.

Players on offense include:

- Quarterback (QB)
- Running Back (RB)
  - Halfback (HB) or Tailback (TB)
  - Fullback (FB)
- Wide Receiver (WR)
- Tight End (TE)
- Offensive Linemen (OL), part of the "Offensive Line"
  - Center (C)
  - Offensive Guard (OG)
  - Offensive Tackle (OT)

The quarterback is the most important player on the offense. They help lead the team down the field. The quarterback receives the ball from the Center at the beginning of the play, and they can either hand the ball off (typically to a Running Back or Fullback), pass the ball (typically to a Wide Receiver or

Tight End), or run the ball. Quarterbacks tend to have a strong arm for throwing the ball far and accurately. Some quarterbacks are fast and are considered "dual threats" to pass or run.

Running Backs take a hand-off from the Quarterback to execute a running play (i.e., a rush). They may also catch short passes from the Quarterback or help protect (i.e., block for) the Quarterback from the defensive players who are trying to tackle the Quarterback. Halfbacks and Tailbacks tend to be quick and agile. Fullbacks tend to be strong and powerful.

Wide Receivers catch passes from the Quarterback to execute a passing play. On running plays, they provide protection for the player running the ball (e.g., the Running Back) so the ball carrier can get as far as possible without being tackled. Wide receivers tend to be tall, fast, have good hands (can catch the ball well), and can jump high.

Tight Ends block for running and passing plays, and they catch passes from the Quarterback. Tight ends tend to be strong and have good hands.

Offensive Linemen block for running and passing plays. On passing plays, they provide protection for the Quarterback so the Quarterback has time to pass the ball without being tackled. On running plays, they provide protection for the player running the ball (e.g., the Running Back) so the ball carrier can get as far as possible without being tackled. Offensive Linemen tend to be large so they can provide adequate protection for the Quarterback and Running Back.

2.1.2.3 Defense

The defense is on the field when the team does not have the ball (i.e., when the opposing team has the ball).

Players on defense include:

- Defensive Linemen (DL), part of the "Defensive Line"
  - Defensive End (DE)
  - Defensive Tackle (DT)
- Linebacker (LB)
  - Middle (or Inside) Linebacker (MLB)
  - Outside Linebacker (OLB)
- Defensive Back (DB), part of the "Secondary"
  - Cornerback (CB)
  - Safety (S)
    - Free Safety (FS)
    - Strong Safety (SS)

The players on the defense attempt to tackle the offensive players for as short of gains as possible and attempt to prevent completed passes.

On passing plays, Defensive Linemen try to apply pressure to the Quarterback and try to tackle the Quarterback behind the line of scrimmage before the Quarterback can throw the ball (i.e., a sack). On rushing plays, Defensive Linemen try to tackle the ball carrier to prevent the ball carrier from advancing the ball (i.e., gaining yards). Defensive Linemen tend to be large yet quick so they can apply pressure to the Quarterback.

Linebackers are versatile in that, on a given play, they may attempt to a) "blitz" to sack the Quarterback, b) stop the Running Back, or c) prevent a completed pass. Linebackers tend to be strong yet agile.

Defensive Backs are specialist pass defenders. The main role of Cornerbacks is to cover the Wide Receivers. Safeties serve as the last line of defense for longer passes. Defensive Backs tend to be quick and agile.

2.1.2.4 Special Teams

The special teams involves specialist players who are on the field during all kicking plays including kickoffs, field goals, and punts.

Players on special teams include:

- Kicker (K)
- Punter (P)
- Holder
- Long Snapper
- Punt Returner
- Kick Returner
- and other players intended to block for or to tackle the ball carrier

On a field goal attempt, the Long Snapper snaps the ball to the Holder, who holds the ball for the Kicker. The Kicker attempts field goals and, during kickoffs, kicks the ball to the opposing team. During kickoffs, the Kick Returner catches the kicked ball and returns it for as many yards as possible. During a punt play, the Long Snapper snaps the ball to the Punter who kicks (i.e., punts) the ball to the opposing team. The Punt Returner catches the punted ball and returns it for as many yards as possible.

2.1.3 The Field

The football field is rectangular and is 120 yards long and 53 1/3 yards wide (109.73 m x 48.77 m).[2] At each end of the 120-yard field is a team's end zone.

[2] One yard is equal to three feet. A yard is just smaller than a meter (0.9144 meters).

Each end zone is 10 yards long (9.14 m). Thus, the distance from one end zone to the other end zone is 100 yards (91.44 m). Behind each end zone is a field goal post. A diagram of a football field is depicted in Figure 2.2.

Figure 2.2 A Diagram of a Football Field. The yard markers depict the distance from the nearest end zone. The orange shaded area is called the "red zone", where chances of scoring points are highest. The original figure was modified to depict field goal posts. (Figure retrieved from https://commons.wikimedia.org/wiki/File:American_football_field.svg)

2.1.4 The Gameplay

At the beginning of the game, there is a coin flip to determine which teams receives the ball first and which team takes which side of the field. During the kickoff, the kicking team kicks the ball to the receiving team, who has the option to return the kick. The offense starts their possession at the 25 yard line—if there is no return (i.e., a touchback)—or wherever the kick returner is tackled or goes out of bounds.

The team with the ball (i.e., the offense) has four opportunities ("downs") to advance the ball (i.e., gain) 10 yards. A team can advance the ball either by running it or by throwing (i.e., passing) and catching it. At the end of a rushing play, the ball advances to wherever the ball carrier is tackled or goes out of bounds (i.e., wherever the player is "down"). At the end of a passing play, if the thrown ball is caught (i.e., a completed pass), the ball advances to wherever the ball carrier is tackled or goes out of bounds. If the thrown ball is not caught in bounds before the ball hits the ground (i.e., an incomplete pass), the ball does not advance. Wherever the ball is advanced to dictates where the next play begins. The yard position on the field where the next play takes place from is known as the "line of scrimmage". Neither team can cross the line the line of scrimmage until the next play begins. To begin the play, the ball is placed on the line of scrimmage and the Center gives (or "snaps") the ball to the Quarterback.

If the team advances the ball 10 or more yards within four downs, the team receives a "first down" and is awarded a new set of downs—four more downs to advance the ball 10 more yards. If the team advances the ball all the way to the other team's end zone, they score a touchdown. If the team fails to advance the ball 10 or more yards within four downs, the team loses the ball, and the other team takes possession at that spot on the field. There are risks of giving the other team the ball with a short distance to score. Thus, on fourth down, instead of trying to advance the ball for a first down, a team may choose to kick a field goal—to get points—or to punt.

A field goal involves a kicker kicking the ball with an intent to kick the ball through the field goal posts ("uprights"). To score points by making a field goal, the kicked ball must go between the uprights (extended vertically) and over the cross bar.

Punting involves a punter kicking the ball to the other team with an intent to give their opponent worse field position, thus making it harder for the other team to score. The punting team tries to pin the opponent as close as possible to the opponent's end zone (i.e., as far as possible from the own team's end zone), so they have a longer distance to go to score a touchdown.

There are multiple ways that ball possession can switch from the offense to the other team. After scoring a touchdown, field goal, or safety, there is a kickoff, in which the scoring team kicks the ball to the opponent. Another

way that the ball switches possession to the other team is if the team commits a turnover. The defense can force a turnover by an interception, fumble recovery, or turnover on downs. A turnover due to an interception occurs when a defensive player catches the quarterback's pass. A turnover due to a fumble recovery occurs when an offensive player, who had possession of the ball, loses the ball before being down or scoring a touchdown and the ball is recovered by the opponent. A turnover on downs occurs when the team attempts on fourth down to achieve the remainder of the needed 10 yards to go but fails.

Other football-related situations include tackles for loss and sacks. A tackle for loss occurs when a ball carrier is tackled behind the line of scrimmage. A sack occurs when a Quarterback is tackled with the ball behind the line of scrimmage. A pass defended occurs when a defensive player knocks down the ball in the air so that the indended receiver cannot catch the ball.

2.1.5 The Scoring

The goal of the team with the ball (i.e., the offense) is to score points. It can do this by either advancing the ball into the other team's end zone (6 points) or by kicking a field goal (3 points). Advancing the ball in the other team's end zone is called a touchdown. After a touchdown, the offense chooses to attempt either a point-after-touchdown (PAT) or a two-point conversion. A PAT is a short kick attempt from the 15-yard line (i.e., 15 yards away from the end zone) that, if it goes through the goal posts ("uprights") and over the cross bar, is worth 1 point. A two-point conversion is a single-scoring opportunity from the 3-yard line (i.e., 3 yards away from the end zone). If the offense scores (i.e., advances the ball into the end zone) from the 3-yard line, the team is awarded 2 points.

A team can kick a field goal from any distance as long as the kick goes through the goal posts. The current record for the longest field goal is 66 yards (by Justin Tucker in 2021).

A safety occurs when the offense is tackled with the ball in their own end zone. When a safety occurs, the opposing team (i.e., defense) is awarded two points and the ball.

2.1.6 Glossary of Terms

- running play ("run") or rushing play (or "rush")—the attempt by an offensive player, typically the Running Back or Quarterback, to advance the ball "on the ground" by running it—not by passing it forward
- passing play (or "pass")—the attempt by an offensive player, typically the Quarterback, to advance the ball by throwing it forward to an offensive player

- passing attempt—the attempt to advance the ball by passing it (i.e., a thrown pass)
- rushing attempt—the attempt to advance the ball by running it
- passing completion—a thrown pass that is succesfully caught by an offensive player
- passing incompletion—a thrown pass that is not caught by an offensive player
- passing yards—the distance (in yards) the player advanced the ball by throwing it
- rushing yards—the distance (in yards) the player advanced the ball by running it
- receving yards—the distance (in yards) the player advanced the ball by catching thrown passes and then running with it further upfield
- kick/punt return yards—the distance (in yards) the player advanced the ball by returning kicks or punts
- turnover return yards—the distance (in yards) the player advanced the ball by returning turnovers
- reception—a pass that is caught by the offensive player
- touchdown—advancing the ball into the opponent's end zone either by a) throwing a completed pass that ends up in the end zone, b) running it into the end zone, c) catching it in the end zone, or d) catching it and then running it into the end zone
- passing touchdown—advancing the ball into the opponent's end zone either by throwing a completed pass that ends up in the end zone
- rushing touchdown—advancing the ball into the opponent's end zone either by running it into the end zone
- receiving touchdown—advancing the ball into the opponent's end zone either by catching it in the end zone or by catching it and then running it into the end zone
- kick/punt return touchdown—advancing the ball into the opponent's end zone when returning a kick or punt
- turnover return touchdown—advancing the ball into the opponent's end zone when returning a turnover (i.e., interception or fumble)
- two-point conversion—a single-scoring opportunity from the 3-yard line (i.e., 3 yards away from the end zone) that is an option given to a team that scores a touchdown; if the offense scores (i.e., advances the ball into the end zone) from the 3-yard line, the team is awarded 2 points
- block—when the defense/special teams blocks a kick or field goal by hitting the ball just after it is kicked to prevent the ball from going far
- kickoff—the kicking team kicks the ball to the receiving team, who has the option to return the kick
- field goal—a kicker kicks the ball with an intent to kick the ball through the field goal posts ("uprights"). To score points by making a field goal, the kicked ball must go between the uprights (extended vertically) and over the cross bar. If the field goal attempt is successful, the team gains 3 points.

- point after touchdown (PAT)—a short kick attempt from the 15-yard line (i.e., 15 yards away from the end zone) that, if it goes through the goal posts ("uprights") and over the cross bar, is worth 1 point
- extra point returned—if the defense/special teams returns the ball into the opponent's end zone during a point after touchdown (PAT) attempt, it is worth 2 points
- punt—a punter kicks the ball to the other team with an intent to give their opponent worse field position, thus making it harder for the other team to score
- fumble lost—when an offensive player, who had possession of the ball, loses the ball before being down or scoring a touchdown and the ball is recovered by the opponent
- fumble forced—when a defensive player knocks the ball out of the hands of an offensive player, who had possession of the ball
- fumble recovery—when a defensive player recovers a fumble by the opponent
- interception—when a defensive player catches a pass from an offensive player
- tackle—when a player brings down the ball carrier
- tackle solo—when a player is the main tackler (i.e., the primary player to bring down the ball carrier)
- tackle assist—when a player is one of two or more players who, together, bring down the ball carrier
- tackle for loss—when an offensive player is tackled with the ball behind the line of scrimmage
- sack—when a Quarterback is tackled with the ball behind the line of scrimmage
- pass defended—when a defensive player knocks down the ball in the air so that the indended receiver cannot catch the ball
- safety—when the offense is tackled with the ball in their own end zone

2.2 Fantasy Football

2.2.1 Overview of Fantasy Football

Fantasy football is one of the most widely played games in the history of games. It is estimated that around 62 million people play fantasy sports[3], of whom around 29 million play fantasy football.[4] As noted in the Introduction[5],

[3] https://thefsga.org/industry-demographics/ (archived at https://perma.cc/9PB8-ZDJJ)

[4] https://www.statista.com/topics/10895/fantasy-sports-in-the-us/ (archived at https://perma.cc/8YSN-UUNT)

[5] intro.qmd

fantasy football is an online game where participants assemble (i.e., "draft") imaginary teams composed of real-life National Football League (NFL) players.[6] The participants are in charge of managing and making strategic decisions for their imaginary team to have the best possible team that will score the most points. Thus, the participants are called "managers". Managers make decisions such as selecting which players to draft, selecting which players to play (i.e., "start") on a weekly basis, identifying players to pick up from the remaining pool of available players (i.e., waiver wire), and making trades with other teams.

There are variety of types of fantasy football leagues. In standard re-draft leagues, managers re-draft players each season. In keeper leagues, managers are allowed to keep one or more players from one season to the next, possibly for some cost (e.g., toward a keeper cap, loss of future draft pick). In dynasty leagues, managers act like a general manager and keep most of their roster from year to year. They may involve player contracts, salary caps, and free agent drafts. In best ball leagues, the manager's best possible lineup (in terms of the highest-scoring players for the necessary roster positions) are automatically selected for that week's lineup.

Fantasy football relies heavily on prediction—trying to predict which players will perform best and selecting them accordingly.

2.2.2 The Fantasy League

A fantasy football "league" is composed of various imaginary (i.e., "fantasy") teams—and their associated manager. In the fantasy league, the managers' fantasy teams play against each other. A fantasy league is commonly composed of 8, 10, or 12 fantasy teams, but leagues can have more or fewer teams.

2.2.3 The Roster of a Fantasy Team

On a given roster, a manager has a "starting lineup" and a "bench". Each week, the manager decides which players on their roster to put in the starting lineup, and which to keep on the bench. In many leagues, a starting lineup is composed of offensive players, a kicker, and defense/special teams:

Offensive players:

[6] Fantasy leagues are also available for baseball[7], basketball[8], and many other sports.

Table 2.1 Offensive Players in the Starting Lineup

Position	Typical Number of Players in Starting Lineup
Quarterback (QB)	1
Running Back (RB)	2
Wide Receiver (WR)	2
Tight End (TE)	1
Flex Position	1

A "flex position" is a flexible position that can involve a player from various positions: e.g., a Running Back, Wide Receiver, or Tight End.

Kickers:

- one Kicker (K)

Defense/Special Teams:

- one Team Defense (DST/D/DEF) or multiple Individual Defensive Players (IDP)

2.2.4 Scoring

2.2.4.1 Scoring Overview

In the game of fantasy football, managers accumulate points on a weekly basis based on players' actual statistical performances in NFL games. Managers receive points for only those players who are on their starting lineup (not players on their bench). In a standard league, a manager's goal is to outscore their opponent each week to win matches and ultimately claim victory in the league. At the end of the regular season, many leagues have a playoffs to determine the league champion. In total points leagues, the league champion is determined by how many points they score throughout the entire season, rather than based on weekly matchups and playoffs.

Scoring settings can differ from league to league. Below are common scoring settings for fantasy leagues.

2.2.4.2 Offensive Players

Table 2.2 Common Scoring Settings for Offensive Players

Statistical category	Points
Rushing or receiving TD	6
Returning a kick or punt for a TD	6
Returning or recovering a fumble for a TD	6
Passing TD	4
Passing INT	−2
Fumble lost	−2
Rushing, passing, or receiving 2-point conversion	2
Rushing or receiving yards	1 point per 10 yards
Passing yards	1 point per 25 yards

Note: "TD" = touchdown; "INT" = interception

Other common (but not necessarily standard) statistical categories include:

- receptions (called "point per reception" [PPR] leagues)
- return yards
- passing attempts
- rushing attempts

2.2.4.3 Kickers

Table 2.3 Common Scoring Settings for Kickers

Statistical category	Points
FG made: 50+ yards	5
FG made: 40–49 yards	4
FG made: 39 yards or less	3
Rushing, passing, or receiving 2-point conversion	2
Point after touchdown attempt made	1
Point after touchdown attempt missed	−1
Missed FG: 0–39 yards	−2
Missed FG: 40–49 yards	−1

Note: "FG" = field goal

2.2.4.4 Team Defense/Special Teams

Table 2.4 Common Scoring Settings for Team Defense/Special Teams

Statistical category	Points
Defensive or special teams TD	6
Interception	2
Fumble recovery	2
Blocked punt, PAT, or FG	2
Safety	2
Sack	1

Note: "TD" = touchdown; "PAT" = point after touchdown; "FG" = field goal

2.2.4.5 Individual Defensive Players

Table 2.5 Common Scoring Settings for Individual Defensive Players

Statistical category	Points
Tackle solo	1
Tackle assist	0.5
Tackle for loss	1
Sack	2
Interception	4
Fumble forced	2
Fumble recovery	2
TD	6
Safety	2
Pass defended	1
Blocked kick	2
Extra point returned	2

Note: "TD" = touchdown

Other common (but not necessarily standard) statistical categories include:

- turnover return yards

2.2.4.6 Common Scoring Abbreviations

- "TD" = touchdown

- "INT" = interception
- "yds" = yards
- "ATT" = attempts
- "2-pt conversion" = two-point conversion
- "FG" = field goal
- "PAT" = point after touchdown (i.e., extra point/point after attempt)

3

Getting Started with R for Data Analysis

The book uses R for statistical analyses (http://www.r-project.org). R is a free software environment; you can download it at no charge here: https://cran.r-project.org.

3.1 Learning R

Here are a various resources for learning R:

- Intro to R: https://www.statmethods.net
- Video training courses in R skills: https://www.pluralsight.com/search?q=R
- Browse the Cookbook for R to find solutions to common tasks and problems: http://www.cookbook-r.com
- Browse the R Graph Gallery to find examples of various graphs: https://r-graph-gallery.com
- Free Codeacademy course on R: https://www.codecademy.com/learn/learn-r
- Free Coursera courses on R: https://www.coursera.org/search?query=R
- Watch these videos from Coursera: https://blog.revolutionanalytics.com/2012/12/coursera-videos.html
- Posit/Rstudio Webinars: https://posit.co/resources/videos/
- UCLA Stats Website: https://stats.idre.ucla.edu/r/
- Introduction to R course on Datacamp: https://www.datacamp.com/courses/free-introduction-to-r
- Teaching R in a Kinder, Gentler, More Effective Manner: https://github.com/matloff/TidyverseSkeptic
- Learn R interactively with swirl: https://swirlstats.com
- Use the learnr package: https://rstudio.github.io/learnr/
- Resources for learning tidyverse, which is a collection of R packages for data management: https://www.tidyverse.org/learn/

- You will sometimes find relevant articles on `R-bloggers`: https://www.r-bloggers.com

3.2 Getting Help with R

If you have R questions, you can ask them in a number of places:

- Forums:
  - `Posit`: https://forum.posit.co
  - `StackOverflow`: https://stackoverflow.com/questions/tagged/r
  - `Reddit`: https://www.reddit.com/r/rstats/
- The R mailing list: https://stat.ethz.ch/mailman/listinfo/r-help

The following article provides additional resources and good guidance: https://www.r-bloggers.com/where-to-get-help-with-your-r-question/.

When posting a question on forums or mailing lists, keep a few things in mind:

- Read the posting guidelines before posting!
- Be respectful of other people and their time. R is free software. People are offering their free time to help. They are under no obligation to help you. If you are disrespectful or act like they owe you anything, you will rub people the wrong way and will be less likely to get help.
- Provide a minimal, reproducible example. Providing a minimal, reproducible example can be crucial for getting a helpful response. By going to the trouble of creating a minimal, reproducible example and identifying the minimum conditions necessary to reproduce the issue, you will often figure out how to resolve it. Here are guidelines on providing a minimal, reproducible example: https://stackoverflow.com/help/minimal-reproducible-example (archived at https://perma.cc/6NUB-UTYF). Here are a good example and guidelines for providing a minimal, reproducible example in R: https://stackoverflow.com/a/5963610 (archived at https://perma.cc/PC9L-DQZG). My strong recommendation is to provide a `reprex` whenever possible: https://reprex.tidyverse.org.

3.3 Initial Setup

To get started, follow the following steps:

1. Install `R`: https://cran.r-project.org
2. Install `RStudio Desktop`: https://posit.co/download/rstudio-desktop
3. After installing `RStudio`, open `RStudio` and run the following code in the console to install several key `R` packages:

```
install.packages(
  c("petersenlab","remotes","nflreadr","nflfastR","nfl4th","nflplotR",
  "gsisdecoder","progressr","lubridate","tidyverse","psych"))
```

4. Some necessary packages, including the `ffanalytics` package, are hosted in GitHub and need to be installed using the following code (after installing the `remotes` package above):

```
remotes::install_github("FantasyFootballAnalytics/ffanalytics")
```

Note 1: If you are in Dr. Petersen's class

If you are in Dr. Petersen's class, also perform the following steps:

1. Download and install `git`: https://git-scm.com/downloads
2. Set up a free account on GitHub.com[a].
3. Download and install `GitHub Desktop`: https://desktop.github.com
4. Make sure you are logged into your GitHub account on GitHub.com[b].
5. Go to the following GitHub repository: https://github.com/isaactpetersen/QuartoBlogFantasyFootball and complete the following steps:
   1. Click "Use this Template" (in the top right of the screen) > "Create a new repository"
   2. Make sure the checkbox is selected for the following option: "Include all branches"
   3. Make sure your Owner account is selected
   4. Specify the repository name to whatever you want, such as `FantasyFootballBlog`
   5. Type a brief description, such as `Files for my fantasy football blog`
   6. Keep the repository public (this is necessary for generating your blog)
   7. Select "Create repository"
6. After creating the new repository, make sure you are on the page of of your new repository and complete the following steps:

1. Click "Settings" (in the top of the screen)
2. Click "Actions" (in the left sidebar) > "General"
3. Make sure the following are selected:
   - "Read and write permissions" (under "Workflow permissions")
   - "Allow GitHub Actions to create and approve pull requests"
   - then click "Save"
4. Click "Pages" (in the left sidebar)
5. Make sure the following are selected:
   - "Deploy from a branch" (under "Source")
   - "gh-pages/(root)" (under "Branch")
   - then click "Save"

7. Clone the repository to your local computer by clicking "Code" > "Open with GitHub Desktop", select the folder where you want the repository to be saved on your local computer, and click "Clone"

[a] https://github.com
[b] https://github.com

3.4 Installing Packages

You can install R packages using the following syntax:

```
install.packages("INSERT_PACKAGE_NAME_HERE")
```

For instance, you can use the following code to install the `tidyverse` package:

```
install.packages("tidyverse")
```

3.5 Load Packages

```
library("tidyverse")
```

3.6 Using Functions and Arguments

You can learn about a particular function and its arguments by entering a question mark before the name of the function:

```
?NAME_OF_FUNCTION()
```

Below, we provide examples for how to learn about and use functions and arguments, by using the `seq()` function as an example. The `seq()` function creates a sequence of numbers. To learn about the `seq()` function, which creates a sequence of numbers, you can execute the following command:

```
?seq()
```

This is what the documentation shows for the `seq()` function in the `Usage` section:

```
seq(
  from = 1,
  to = 1,
  by = ((to - from)/(length.out - 1)),
  length.out = NULL,
  along.with = NULL,
  ...)
```

Based on this information, we know that the `seq()` function takes the following arguments:

- `from`
- `to`
- `by`
- `length.out`
- `along.with`
- `...`

The arguments have default values that are used if the user does not specify values for the arguments. The default values are provided in the `Usage` section and are in Table 3.1:

Table 3.1 Arguments and defaults for the `seq()` function. Arguments with a default of `NULL` are not used unless a value is provided by the user.

Argument	Default Value for Argument
`from`	`1`
`to`	`1`
`by`	`((to - from)/(length.out - 1))`
`length.out`	`NULL`
`along.with`	`NULL`

What each argument represents (i.e., the meaning of `from`, `to`, `by`, etc.) is provided in the `Arguments` section of the documentation. You can specify a function and its arguments either by providing values for each argument in the order indicated by the function, or by naming its arguments.

Here is an example of providing values to the arguments in the order indicated by the function, to create a sequence of numbers from 1 to 9:

```
seq(1, 9)
```

```
[1] 1 2 3 4 5 6 7 8 9
```

Here is an example of providing values to the arguments by naming its arguments:

```
seq(
  from = 1,
  to = 9,
  by = 1)
```

```
[1] 1 2 3 4 5 6 7 8 9
```

If you provide values to arguments by naming the arguments, you can reorder the arguments and get the same answer:

```
seq(
  by = 1,
  to = 9,
  from = 1)
```

```
[1] 1 2 3 4 5 6 7 8 9
```

There are various combinations of arguments that one could use to obtain the same result. For instance, here is code to generate a sequence from 1 to 9 by 2:

```
seq(
  from = 1,
  to = 9,
  by = 2)
```

```
[1] 1 3 5 7 9
```

Or, alternatively, you could specify the length of the desired sequence (5 values):

```
seq(
  from = 1,
  to = 9,
  length.out = 5)
```

```
[1] 1 3 5 7 9
```

If you want to generate a series with decimal values, you could specify a long desired sequence of 81 values:

```
seq(
  from = 1,
  to = 9,
  length.out = 81)
```

```
 [1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8
[20] 2.9 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1 4.2 4.3 4.4 4.5 4.6 4.7
[39] 4.8 4.9 5.0 5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 6.0 6.1 6.2 6.3 6.4 6.5 6.6
[58] 6.7 6.8 6.9 7.0 7.1 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9 8.0 8.1 8.2 8.3 8.4 8.5
[77] 8.6 8.7 8.8 8.9 9.0
```

This is equivalent to specifying a sequence from 1 to 9 by 0.1:

```
seq(
  from = 1,
  to = 9,
  by = 0.1)
```

```
 [1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8
[20] 2.9 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1 4.2 4.3 4.4 4.5 4.6 4.7
[39] 4.8 4.9 5.0 5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 6.0 6.1 6.2 6.3 6.4 6.5 6.6
[58] 6.7 6.8 6.9 7.0 7.1 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9 8.0 8.1 8.2 8.3 8.4 8.5
[77] 8.6 8.7 8.8 8.9 9.0
```

Hopefully, that provides an example for how to learn about a particular function, its arguments, and how to use them.

3.7 Create a Vector

A vector is a series of elements that can be numeric or character. It has one dimension (length). To create a vector, use the `c()` to combine elements into a vector:

```
exampleVector <- c(40, 30, 24, 20, 18, 23, 27, 32, 26, 23, NA, 37)
```

3.8 Create a Data Frame

A data frame has two dimensions: rows and columns. Here is an example of creating a data frame:

```
players <- data.frame(
  ID = 1:12,
  name = c(
    "Ken Cussion",
    "Ben Sacked",
    "Chuck Downfield",
    "Ron Ingback",
    "Rhonda Ball",
    "Hugo Long",
    "Lionel Scrimmage",
    "Drew Blood",
    "Chase Emdown",
    "Justin Time",
    "Spike D'Ball",
    "Isac Ulooz"),
```

```
  position = c("QB","QB","QB","RB","RB","WR","WR","WR","WR","TE","TE","LB"),
  age = c(40, 30, 24, 20, 18, 23, 27, 32, 26, 23, NA, 37)
  )

fantasyPoints <- data.frame(
  ID = c(2, 7, 13, 14),
  fantasyPoints = c(250, 170, 65, 15)
)

fantasyPoints_weekly = expand.grid(
  ID = 1:12,
  season = c(2022, 2023),
  week = 1:17
)

set.seed(52242)
fantasyPoints_weekly$fantasyPoints <- sample(
  0:35,
  size = nrow(fantasyPoints_weekly),
  replace = TRUE
)
```

3.9 Create a List

A list can store multiple data frames in one object:

```
exampleList <- list(players, fantasyPoints, fantasyPoints_weekly)
```

3.10 Load a Data Frame

```
load(file = "./data/nfl_players.RData")
```

3.11 Variable Names

To see the names of variables in a data frame, use the following syntax:

```
names(nfl_players)
```

```
 [1] "gsis_id"                  "status"
 [3] "display_name"             "first_name"
 [5] "last_name"                "esb_id"
 [7] "birth_date"               "college_name"
 [9] "position_group"           "position"
[11] "jersey_number"            "height"
[13] "weight"                   "years_of_experience"
[15] "team_abbr"                "team_seq"
[17] "current_team_id"          "football_name"
[19] "entry_year"               "rookie_year"
[21] "draft_club"               "draft_number"
[23] "college_conference"       "status_description_abbr"
[25] "status_short_description" "gsis_it_id"
[27] "short_name"               "smart_id"
[29] "headshot"                 "suffix"
[31] "uniform_number"           "draft_round"
[33] "season"
```

```
names(players)
```

```
[1] "ID"       "name"     "position" "age"
```

```
names(fantasyPoints)
```

```
[1] "ID"            "fantasyPoints"
```

3.12 Logical Operators

3.12.1 Is Equal To: ==

```
players$position == "RB"
```

```
 [1] FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

3.12.2 Is Not Equal To: `!=`

```
players$position != "RB"
```

```
 [1]  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

3.12.3 Is Greater Than: `>`

```
players$age > 30
```

```
 [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE   NA  TRUE
```

3.12.4 Is Less Than: `<`

```
players$age < 30
```

```
 [1] FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE   NA FALSE
```

3.12.5 Is Greater Than or Equal To: `>=`

```
players$age >= 30
```

```
 [1]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE   NA  TRUE
```

3.12.6 Is Less Than or Equal To: `<=`

```
players$age <= 30
```

```
 [1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE    NA FALSE
```

3.12.7 Is In a Value of Another Vector: `%in%`

```
players$position %in% c("RB","WR")
```

```
 [1] FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE
```

3.12.8 Is Not In a Value of Another Vector: `!(%in%)`

```
!(players$position %in% c("RB","WR"))
```

```
 [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
```

3.12.9 Is Missing: `is.na()`

```
is.na(players$age)
```

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
```

3.12.10 Is Not Missing: `!is.na()`

```
!is.na(players$age)
```

```
 [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
```

3.12.11 And: `&`

```
players$position == "WR" & players$age > 26
```

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE
```

3.12.12 Or: |

```
players$position == "WR" | players$age > 23
```

```
 [1]  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE    NA  TRUE
```

3.13 Piping

In base R, if you want to perform multiple operations, it is common to either a) nest the operations, or b) save the object at each step.

Below is an example of nested operations:

```
length(names(nfl_players))
```

```
[1] 33
```

Below is an example of saving the intermediate object at each step:

```
variableNames <- names(nfl_players)
variableNames
```

```
 [1] "gsis_id"                  "status"
 [3] "display_name"             "first_name"
 [5] "last_name"                "esb_id"
 [7] "birth_date"               "college_name"
 [9] "position_group"           "position"
[11] "jersey_number"            "height"
[13] "weight"                   "years_of_experience"
[15] "team_abbr"                "team_seq"
[17] "current_team_id"          "football_name"
[19] "entry_year"               "rookie_year"
```

```
[21] "draft_club"               "draft_number"
[23] "college_conference"       "status_description_abbr"
[25] "status_short_description" "gsis_it_id"
[27] "short_name"               "smart_id"
[29] "headshot"                 "suffix"
[31] "uniform_number"           "draft_round"
[33] "season"
```

```
lengthOfVariableNames <- length(variableNames)
lengthOfVariableNames
```

```
[1] 33
```

Code for performing nested operations can be challenging to read. Saving the intermediate object can be a waste of time to do if you are not interested in the intermediate object, and can take up unnecessary memory and computational resources. An alternative approach is to use piping. Piping allows taking the result from one computation and sending it to the next computation, thus allowing a chain of computations without saving the intermediate object at each step.

In base R, you can perform piping with the `|>` expression. In `tidyverse` you can perform piping with the `%>%` expression.

3.13.0.1 Base R

```
nfl_players |>
  names() |>
  length()
```

```
[1] 33
```

3.13.0.2 Tidyverse

```
nfl_players %>%
  names() %>%
  length()
```

```
[1] 33
```

3.14 Subset

To subset a data frame, use brackets to specify the subset of rows and columns to keep, where the value/vector before the comma specifies the rows to keep, and the value/vector after the comma specifies the columns to keep:

```
dataframe[rowsToKeep, columnsToKeep]
```

You can subset by using any of the following:

- numeric indices of the rows/columns to keep (or drop)
- names of the rows/columns to keep (or drop)
- values of `TRUE` and `FALSE` corresponding to which rows/columns to keep

3.14.1 One Variable

To subset one variable, use the following syntax:

```
players$name
```

```
 [1] "Ken Cussion"     "Ben Sacked"      "Chuck Downfield"  "Ron Ingback"
 [5] "Rhonda Ball"     "Hugo Long"       "Lionel Scrimmage" "Drew Blood"
 [9] "Chase Emdown"    "Justin Time"     "Spike D'Ball"     "Isac Ulooz"
```

or:

```
players[,"name"]
```

```
 [1] "Ken Cussion"     "Ben Sacked"      "Chuck Downfield"  "Ron Ingback"
 [5] "Rhonda Ball"     "Hugo Long"       "Lionel Scrimmage" "Drew Blood"
 [9] "Chase Emdown"    "Justin Time"     "Spike D'Ball"     "Isac Ulooz"
```

3.14.2 Particular Rows of One Variable

To subset one variable, use the following syntax:

```
players$name[which(players$position == "RB")]
```

```
[1] "Ron Ingback" "Rhonda Ball"
```

or:

```
players[which(players$position == "RB"), "name"]
```

```
[1] "Ron Ingback" "Rhonda Ball"
```

3.14.3 Particular Columns (Variables)

To subset particular columns/variables, use the following syntax:

3.14.3.1 Base R

```
subsetVars <- c("name","age")

players[,c(2,4)]
```

```
               name age
1       Ken Cussion  40
2        Ben Sacked  30
3   Chuck Downfield  24
4       Ron Ingback  20
5       Rhonda Ball  18
6         Hugo Long  23
7  Lionel Scrimmage  27
8        Drew Blood  32
9      Chase Emdown  26
10      Justin Time  23
11     Spike D'Ball  NA
12       Isac Ulooz  37
```

```
players[,c("name","age")]
```

```
               name age
1       Ken Cussion  40
2        Ben Sacked  30
3   Chuck Downfield  24
4       Ron Ingback  20
5       Rhonda Ball  18
6         Hugo Long  23
7  Lionel Scrimmage  27
8        Drew Blood  32
```

```
9      Chase Emdown  26
10      Justin Time  23
11     Spike D'Ball  NA
12       Isac Ulooz  37
```

```
players[,subsetVars]
```

```
               name age
1       Ken Cussion  40
2        Ben Sacked  30
3   Chuck Downfield  24
4       Ron Ingback  20
5       Rhonda Ball  18
6         Hugo Long  23
7  Lionel Scrimmage  27
8        Drew Blood  32
9      Chase Emdown  26
10      Justin Time  23
11     Spike D'Ball  NA
12       Isac Ulooz  37
```

Or, to drop columns:

```
dropVars <- c("name","age")

players[,-c(2,4)]
```

```
   ID position
1   1       QB
2   2       QB
3   3       QB
4   4       RB
5   5       RB
6   6       WR
7   7       WR
8   8       WR
9   9       WR
10 10       TE
11 11       TE
12 12       LB
```

```
players[,!(names(players) %in% c("name","age"))]
```

```
   ID position
1   1       QB
2   2       QB
3   3       QB
4   4       RB
5   5       RB
6   6       WR
7   7       WR
8   8       WR
9   9       WR
10 10       TE
11 11       TE
12 12       LB
```

```
players[,!(names(players) %in% dropVars)]
```

```
   ID position
1   1       QB
2   2       QB
3   3       QB
4   4       RB
5   5       RB
6   6       WR
7   7       WR
8   8       WR
9   9       WR
10 10       TE
11 11       TE
12 12       LB
```

3.14.3.2 Tidyverse

```
players %>%
  select(name, age)
```

```
               name age
1       Ken Cussion  40
2        Ben Sacked  30
3   Chuck Downfield  24
4       Ron Ingback  20
5       Rhonda Ball  18
6         Hugo Long  23
7  Lionel Scrimmage  27
```

```
8        Drew Blood  32
9      Chase Emdown  26
10      Justin Time  23
11     Spike D'Ball  NA
12       Isac Ulooz  37
```

```r
players %>%
  select(name:age)
```

```
               name position age
1       Ken Cussion       QB  40
2        Ben Sacked       QB  30
3   Chuck Downfield       QB  24
4       Ron Ingback       RB  20
5       Rhonda Ball       RB  18
6         Hugo Long       WR  23
7  Lionel Scrimmage       WR  27
8        Drew Blood       WR  32
9      Chase Emdown       WR  26
10      Justin Time       TE  23
11     Spike D'Ball       TE  NA
12       Isac Ulooz       LB  37
```

```r
players %>%
  select(all_of(subsetVars))
```

```
               name age
1       Ken Cussion  40
2        Ben Sacked  30
3   Chuck Downfield  24
4       Ron Ingback  20
5       Rhonda Ball  18
6         Hugo Long  23
7  Lionel Scrimmage  27
8        Drew Blood  32
9      Chase Emdown  26
10      Justin Time  23
11     Spike D'Ball  NA
12       Isac Ulooz  37
```

Or, to drop columns:

```
players %>%
  select(-name, -age)
```

```
   ID position
1   1       QB
2   2       QB
3   3       QB
4   4       RB
5   5       RB
6   6       WR
7   7       WR
8   8       WR
9   9       WR
10 10       TE
11 11       TE
12 12       LB
```

```
players %>%
  select(-c(name:age))
```

```
   ID
1   1
2   2
3   3
4   4
5   5
6   6
7   7
8   8
9   9
10 10
11 11
12 12
```

```
players %>%
  select(-all_of(dropVars))
```

```
   ID position
1   1       QB
2   2       QB
3   3       QB
4   4       RB
5   5       RB
```

```
6   6       WR
7   7       WR
8   8       WR
9   9       WR
10 10       TE
11 11       TE
12 12       LB
```

3.14.4 Particular Rows

To subset particular rows, use the following syntax:

3.14.4.1 Base R

```r
subsetRows <- c(4,5)

players[c(4,5),]
```

```
  ID        name position age
4  4 Ron Ingback       RB  20
5  5 Rhonda Ball       RB  18
```

```r
players[subsetRows,]
```

```
  ID        name position age
4  4 Ron Ingback       RB  20
5  5 Rhonda Ball       RB  18
```

```r
players[which(players$position == "RB"),]
```

```
  ID        name position age
4  4 Ron Ingback       RB  20
5  5 Rhonda Ball       RB  18
```

3.14.4.2 Tidyverse

```r
players %>%
  filter(position == "WR")
```

```
  ID             name position age
1  6        Hugo Long       WR  23
2  7 Lionel Scrimmage       WR  27
3  8       Drew Blood       WR  32
4  9     Chase Emdown       WR  26
```

```
players %>%
  filter(position == "WR", age <= 26)
```

```
  ID         name position age
1  6    Hugo Long       WR  23
2  9 Chase Emdown       WR  26
```

```
players %>%
  filter(position == "WR" | age >= 26)
```

```
  ID             name position age
1  1      Ken Cussion       QB  40
2  2       Ben Sacked       QB  30
3  6        Hugo Long       WR  23
4  7 Lionel Scrimmage       WR  27
5  8       Drew Blood       WR  32
6  9     Chase Emdown       WR  26
7 12       Isac Ulooz       LB  37
```

3.14.5 Particular Rows and Columns

To subset particular rows and columns, use the following syntax:

3.14.5.1 Base R

```
players[c(4,5), c(2,4)]
```

```
         name age
4 Ron Ingback  20
5 Rhonda Ball  18
```

```
players[subsetRows, subsetVars]
```

```
         name age
4 Ron Ingback  20
5 Rhonda Ball  18
```

```
players[which(players$position == "RB"), subsetVars]
```

```
         name age
4 Ron Ingback  20
5 Rhonda Ball  18
```

3.14.5.2 Tidyverse

```
players %>%
  filter(position == "RB") %>%
  select(all_of(subsetVars))
```

```
         name age
1 Ron Ingback  20
2 Rhonda Ball  18
```

3.15 View Data

3.15.1 All Data

To view data, use the following syntax:

```
View(players)
```

3.15.2 First 6 Rows/Elements

To view only the first six rows (if a data frame) or elements (if a vector), use the following syntax:

```
head(nfl_players)
```

```
# A tibble: 6 x 33
  gsis_id   status display_name    first_name last_name esb_id   birth_date
  <chr>      <chr>  <chr>            <chr>      <chr>     <chr>     <chr>
1 00-0004866 RET    'Omar Ellison    'Omar      Ellison   ELL711319 <NA>
2 00-0032889 ACT   A'Shawn Robinson A'Shawn   Robinson  ROB367960 1995-03-21
```

```
3 00-0037845 ACT    A.J. Arcuri      A.J.       Arcuri    ARC716900 <NA>
4 00-0039793 ACT    A.J. Barner      A.J.       Barner    <NA>      <NA>
5 00-0030228 RES   A.J. Bouye     Arlandus  Bouye    BOU651714 1991-08-16
6 00-0035676 ACT   A.J. Brown     Arthur    Brown    BRO413223 1997-06-30
# i 26 more variables: college_name <chr>, position_group <chr>,
#   position <chr>, jersey_number <int>, height <dbl>, weight <int>,
#   years_of_experience <chr>, team_abbr <chr>, team_seq <int>,
#   current_team_id <chr>, football_name <chr>, entry_year <int>,
#   rookie_year <int>, draft_club <chr>, draft_number <int>,
#   college_conference <chr>, status_description_abbr <chr>,
#  status_short_description <chr>, gsis_it_id <int>, short_name <chr>, ...
```

```r
head(nfl_players$display_name)
```

```
[1] "'Omar Ellison"   "A'Shawn Robinson" "A.J. Arcuri"     "A.J. Barner"
[5] "A.J. Bouye"       "A.J. Brown"
```

3.16 Data Characteristics

3.16.1 Data Structure

```r
str(nfl_players)
```

```
nflvrs_d [20,721 x 33] (S3: nflverse_data/tbl_df/tbl/data.table/data.frame)
 $ gsis_id            : chr [1:20721] "00-0004866" "00-0032889" "00-0037845" "00-0039793" ...
 $ status             : chr [1:20721] "RET" "ACT" "ACT" "ACT" ...
 $ display_name       : chr [1:20721] "'Omar Ellison" "A'Shawn Robinson" "A.J. Arcuri" "A.J. Barner" .
 $ first_name         : chr [1:20721] "'Omar" "A'Shawn" "A.J." "A.J." ...
 $ last_name          : chr [1:20721] "Ellison" "Robinson" "Arcuri" "Barner" ...
 $ esb_id             : chr [1:20721] "ELL711319" "ROB367960" "ARC716900" NA ...
 $ birth_date         : chr [1:20721] NA "1995-03-21" NA NA ...
 $ college_name       : chr [1:20721] NA "Alabama" "Michigan State" "Michigan" ...
 $ position_group     : chr [1:20721] "WR" "DL" "OL" "TE" ...
 $ position           : chr [1:20721] "WR" "DT" "T" "TE" ...
 $ jersey_number      : int [1:20721] 84 94 61 88 24 11 60 6 81 63 ...
 $ height             : num [1:20721] 73 76 79 78 72 72 75 76 69 76 ...
 $ weight             : int [1:20721] 200 330 320 251 191 226 325 220 190 280 ...
 $ years_of_experience : chr [1:20721] "2" "8" "2" NA ...
 $ team_abbr          : chr [1:20721] "LAC" "CAR" "LA" "SEA" ...
```

```
 $ team_seq                : int [1:20721] NA 1 NA NA 1 1 1 1 NA NA ...
 $ current_team_id       : chr [1:20721] "4400" "0750" "2510" "4600" ...
 $ football_name          : chr [1:20721] NA "A'Shawn" "A.J." "A.J." ...
 $ entry_year          : int [1:20721] NA 2016 2022 2024 2013 2019 2015 2019 NA NA ...
 $ rookie_year         : int [1:20721] NA 2016 2022 2024 2013 2019 2015 2019 NA NA ...
 $ draft_club              : chr [1:20721] NA "DET" "LA" "SEA" ...
 $ draft_number         : int [1:20721] NA 46 261 121 NA 51 67 NA NA NA ...
 $ college_conference     : chr [1:20721] NA "Southeastern Conference" "Big Ten Conference" "Big Ten Con
 $ status_description_abbr : chr [1:20721] NA "A01" "A01" "A01" ...
 $ status_short_description: chr [1:20721] NA "Active" "Active" "Active" ...
 $ gsis_it_id          : int [1:20721] NA 43335 54726 57242 40688 47834 42410 48335 NA NA ...
 $ short_name          : chr [1:20721] NA "A.Robinson" "A.Arcuri" "A.Barner" ...
 $ smart_id           : chr [1:20721] "3200454c-4c71-1319-728e-d49d3d236f8f" "3200524f-4236-7960-bf20
 $ headshot           : chr [1:20721] NA "https://static.www.nfl.com/image/private/f_auto,q_auto/leag
 $ suffix                  : chr [1:20721] NA NA NA NA ...
 $ uniform_number          : chr [1:20721] NA "94" "61" "88" ...
 $ draft_round             : chr [1:20721] NA NA NA NA ...
 $ season               : int [1:20721] NA NA NA NA NA NA NA NA NA NA ...
 - attr(*, "nflverse_type")= chr "players"
 - attr(*, "nflverse_timestamp")= POSIXct[1:1], format: "2024-08-01 01:29:00"
 - attr(*, ".internal.selfref")=<externalptr>
```

3.16.2 Data Dimensions

Number of rows and columns:

```r
dim(nfl_players)
```

```
[1] 20721    33
```

Number of rows:

```r
nrow(nfl_players)
```

```
[1] 20721
```

Number of columns:

```r
ncol(nfl_players)
```

```
[1] 33
```

3.16.3 Number of Elements

```
length(nfl_players$display_name)
```

```
[1] 20721
```

3.16.4 Number of Missing Elements

```
length(nfl_players$college_name[which(is.na(nfl_players$college_name))])
```

```
[1] 12126
```

3.16.5 Number of Non-Missing Elements

```
length(nfl_players$college_name[which(!is.na(nfl_players$college_name))])
```

```
[1] 8595
```

```
length(na.omit(nfl_players$college_name))
```

```
[1] 8595
```

3.17 Create New Variables

To create a new variable, use the following syntax:

```
players$newVar <- NA
```

Here is an example of creating a new variable:

```
players$newVar <- 1:nrow(players)
```

3.18 Recode Variables

Here is an example of recoding a variable:

```
players$oldVar1 <- NA
players$oldVar1[which(players$position == "QB")] <- "quarterback"
players$oldVar1[which(players$position == "RB")] <- "running back"
players$oldVar1[which(players$position == "WR")] <- "wide receiver"
players$oldVar1[which(players$position == "TE")] <- "tight end"

players$oldVar2 <- NA
players$oldVar2[which(players$age < 30)] <- "young"
players$oldVar2[which(players$age >= 30)] <- "old"
```

Recode multiple variables:

```
players %>%
  mutate(across(c(
    oldVar1:oldVar2),
    ~ case_match(
      .,
      c("quarterback","old","running back") ~ 0,
      c("wide receiver","tight end","young") ~ 1)))
```

```
   ID             name position age oldVar1 oldVar2
1   1      Ken Cussion       QB  40       0       0
2   2       Ben Sacked       QB  30       0       0
3   3  Chuck Downfield       QB  24       0       1
4   4      Ron Ingback       RB  20       0       1
5   5      Rhonda Ball       RB  18       0       1
6   6        Hugo Long       WR  23       1       1
7   7 Lionel Scrimmage       WR  27       1       1
8   8       Drew Blood       WR  32       1       0
9   9     Chase Emdown       WR  26       1       1
10 10      Justin Time       TE  23       1       1
11 11     Spike D'Ball       TE  NA       1      NA
12 12       Isac Ulooz       LB  37      NA       0
```

3.19 Rename Variables

```
players <- players %>%
  rename(
    newVar1 = oldVar1,
    newVar2 = oldVar2)
```

Using a vector of variable names:

```
varNamesFrom <- c("oldVar1","oldVar2")
varNamesTo <- c("newVar1","newVar2")

players <- players %>%
  rename_with(~ varNamesTo, all_of(varNamesFrom))
```

3.20 Convert the Types of Variables

One variable:

```
players$factorVar <- factor(players$ID)
players$numericVar <- as.numeric(players$age)
players$integerVar <- as.integer(players$newVar1)
players$characterVar <- as.character(players$newVar2)
```

Multiple variables:

```
players %>%
  mutate(across(c(
    ID,
    age),
    as.numeric))
```

```
  ID          name position age      newVar1 newVar2 factorVar numericVar
1  1     Ken Cussion      QB  40   quarterback     old        1        40
2  2      Ben Sacked      QB  30   quarterback     old        2        30
3  3  Chuck Downfield      QB  24   quarterback   young        3        24
```

```
4   4     Ron Ingback      RB  20  running back   young        4        20
5   5     Rhonda Ball      RB  18  running back   young        5        18
6   6       Hugo Long      WR  23 wide receiver   young        6        23
7   7 Lionel Scrimmage      WR  27 wide receiver   young        7        27
8   8      Drew Blood      WR  32 wide receiver    old        8        32
9   9    Chase Emdown      WR  26 wide receiver   young        9        26
10 10     Justin Time      TE  23     tight end   young       10        23
11 11    Spike D'Ball      TE  NA     tight end    <NA>       11        NA
12 12      Isac Ulooz      LB  37          <NA>    old       12        37
   integerVar characterVar
1          NA          old
2          NA          old
3          NA        young
4          NA        young
5          NA        young
6          NA        young
7          NA        young
8          NA          old
9          NA        young
10         NA        young
11         NA         <NA>
12         NA          old
```

```r
players %>%
  mutate(across(
    age:newVar1,
    as.character))
```

```
  ID          name position  age     newVar1 newVar2 factorVar numericVar
1  1     Ken Cussion      QB   40   quarterback     old        1        40
2  2      Ben Sacked      QB   30   quarterback     old        2        30
3  3  Chuck Downfield      QB   24   quarterback   young        3        24
4  4     Ron Ingback      RB   20  running back   young        4        20
5  5     Rhonda Ball      RB   18  running back   young        5        18
6  6       Hugo Long      WR   23 wide receiver   young        6        23
7  7 Lionel Scrimmage      WR   27 wide receiver   young        7        27
8  8      Drew Blood      WR   32 wide receiver    old        8        32
9  9    Chase Emdown      WR   26 wide receiver   young        9        26
10 10     Justin Time      TE   23     tight end   young       10        23
11 11    Spike D'Ball      TE <NA>     tight end    <NA>       11        NA
12 12      Isac Ulooz      LB   37          <NA>    old       12        37
   integerVar characterVar
1          NA          old
2          NA          old
3          NA        young
```

```
4           NA         young
5           NA         young
6           NA         young
7           NA         young
8           NA           old
9           NA         young
10          NA         young
11          NA          <NA>
12          NA           old
```

```r
players %>%
  mutate(across(where(is.factor), as.character))
```

```
   ID          name position age      newVar1 newVar2 factorVar numericVar
1   1     Ken Cussion      QB  40   quarterback     old        1        40
2   2      Ben Sacked      QB  30   quarterback     old        2        30
3   3  Chuck Downfield     QB  24   quarterback   young        3        24
4   4     Ron Ingback      RB  20  running back   young        4        20
5   5     Rhonda Ball      RB  18  running back   young        5        18
6   6       Hugo Long     WR  23 wide receiver   young        6        23
7   7 Lionel Scrimmage     WR  27 wide receiver   young        7        27
8   8      Drew Blood      WR  32 wide receiver     old        8        32
9   9    Chase Emdown      WR  26 wide receiver   young        9        26
10 10     Justin Time      TE  23     tight end   young       10        23
11 11    Spike D'Ball      TE  NA     tight end    <NA>       11        NA
12 12      Isac Ulooz      LB  37          <NA>     old       12        37
   integerVar characterVar
1          NA           old
2          NA           old
3          NA         young
4          NA         young
5          NA         young
6          NA         young
7          NA         young
8          NA           old
9          NA         young
10         NA         young
11         NA          <NA>
12         NA           old
```

3.21 Merging/Joins

3.21.1 Overview

Merging (also called joining) merges two data objects using a shared set of variables called "keys." The keys are the variable(s) that uniquely identify each row (i.e., they account for the levels of nesting). In some data objects, the key might be the player's identification number (e.g., `player_id`). However, some data objects have multiple keys. For instance, in long form data objects, each participant may have multiple rows corresponding to multiple seasons. In this case, the keys may be `player_id` and `season`—that is, the data are in `player-season` form. If a participant has multiple rows corresponding to seasons and games/weeks, the keys are `player_id`, `season`, and `week`—in this case, the data are in `player-season-week` form. In general, each row should have a value on each of the keys; there should be no missingness in the keys.

To merge two objects, the key(s) that will be used to match the records must be present in both objects. The keys are used to merge the variables in object 1 (`x`) with the variables in object 2 (`y`). Different merge types select different rows to merge.

Note: if the two objects include variables with the same name (apart from the keys), `R` will not know how you want each to appear in the merged object. So, it will add a suffix (e.g., `.x`, `.y`) to each common variable to indicate which object (i.e., object `x` or object `y`) the variable came from, where object `x` is the first object—i.e., the object to which object `y` (the second object) is merged. In general, apart from the keys, you should not include variables with the same name in two objects to be merged. To prevent this, either remove or rename the shared variable in one of the objects, or include the shared variable as a key. However, as described above, you should include it as a key ***only*** if it uniquely identifies each row in terms of levels of nesting.

When two objects are merged that have different formats, the resulting data object inherits the format of the data object that has more levels of nesting. For instance, consider that you want to merge two objects, object A and object B. Object A is in `player` form and object B is in `player-season-week` form. When you merge them, the resulting data object will be in `player-season-week` form.

3.21.2 Data Before Merging

Here are the data in the `players` object:

```r
players
```

```
   ID          name position age      newVar1 newVar2 factorVar numericVar
1   1     Ken Cussion      QB  40   quarterback     old        1        40
2   2      Ben Sacked      QB  30   quarterback     old        2        30
3   3  Chuck Downfield      QB  24   quarterback   young        3        24
4   4     Ron Ingback      RB  20  running back   young        4        20
5   5     Rhonda Ball      RB  18  running back   young        5        18
6   6       Hugo Long      WR  23 wide receiver   young        6        23
7   7 Lionel Scrimmage      WR  27 wide receiver   young        7        27
8   8      Drew Blood      WR  32 wide receiver     old        8        32
9   9    Chase Emdown      WR  26 wide receiver   young        9        26
10 10     Justin Time      TE  23     tight end   young       10        23
11 11    Spike D'Ball      TE  NA     tight end    <NA>       11        NA
12 12      Isac Ulooz      LB  37         <NA>     old       12        37
   integerVar characterVar
1          NA          old
2          NA          old
3          NA        young
4          NA        young
5          NA        young
6          NA        young
7          NA        young
8          NA          old
9          NA        young
10         NA        young
11         NA         <NA>
12         NA          old
```

```r
dim(players)
```

```
[1] 12 10
```

The data are structured in ID form. That is, every row in the dataset is uniquely identified by the variable, `ID`.

Here are the data in the `fantasyPoints` object:

```r
fantasyPoints
```

```
  ID fantasyPoints
1  2           250
2  7           170
3 13            65
4 14            15
```

```
dim(fantasyPoints)
```

```
[1] 4 2
```

3.21.3 Types of Joins

3.21.3.1 Visual Overview of Join Types

Below is a visual that depicts various types of merges/joins. Object `x` is the circle labeled as `x`. Object `y` is the circle labeled as `y`. The area of overlap in the Venn diagram indicates the rows on the keys that are shared between the two objects (e.g., the same `player_id`, `season`, and `week`). The non-overlapping area indicates the rows on the keys that are unique to each object. The shaded blue area indicates which rows (on the keys) are kept in the merged object from each of the two objects, when using each of the merge types. For instance, a left outer join keeps the shared rows and the rows that are unique to object `x`, but it drops the rows that are unique to object `y`.

Figure 3.1 Types of merges/joins

3.21.3.2 Full Outer Join

A full outer join includes all rows in `x` **or** `y`. It returns columns from `x` and `y`. Here is how to merge two data frames using a full outer join (i.e., "full join"):

```r
fullJoinData <- full_join(
  players,
  fantasyPoints,
  by = "ID")

fullJoinData
```

```
   ID          name position age      newVar1 newVar2 factorVar numericVar
1   1     Ken Cussion      QB  40   quarterback    old        1        40
2   2      Ben Sacked      QB  30   quarterback    old        2        30
3   3  Chuck Downfield     QB  24   quarterback  young        3        24
4   4     Ron Ingback      RB  20  running back  young        4        20
5   5     Rhonda Ball      RB  18  running back  young        5        18
6   6       Hugo Long     WR  23 wide receiver  young        6        23
7   7 Lionel Scrimmage     WR  27 wide receiver  young        7        27
8   8      Drew Blood     WR  32 wide receiver    old        8        32
9   9    Chase Emdown     WR  26 wide receiver  young        9        26
10 10     Justin Time      TE  23    tight end  young       10        23
11 11    Spike D'Ball      TE  NA    tight end   <NA>       11        NA
12 12      Isac Ulooz      LB  37         <NA>    old       12        37
13 13           <NA>    <NA>  NA         <NA>   <NA>     <NA>        NA
14 14           <NA>    <NA>  NA         <NA>   <NA>     <NA>        NA
   integerVar characterVar fantasyPoints
1          NA          old            NA
2          NA          old           250
3          NA        young            NA
4          NA        young            NA
5          NA        young            NA
6          NA        young            NA
7          NA        young           170
8          NA          old            NA
9          NA        young            NA
10         NA        young            NA
11         NA         <NA>            NA
12         NA          old            NA
13         NA         <NA>            65
14         NA         <NA>            15
```

```r
dim(fullJoinData)
```

```
[1] 14 11
```

3.21.3.3 Left Outer Join

A left outer join includes all rows in `x`. It returns columns from `x` and `y`. Here is how to merge two data frames using a left outer join ("left join"):

```
leftJoinData <- left_join(
  players,
  fantasyPoints,
  by = "ID")

leftJoinData
```

```
   ID          name position age      newVar1 newVar2 factorVar numericVar
1   1     Ken Cussion      QB  40   quarterback     old        1        40
2   2      Ben Sacked      QB  30   quarterback     old        2        30
3   3  Chuck Downfield     QB  24   quarterback   young        3        24
4   4     Ron Ingback      RB  20  running back   young        4        20
5   5     Rhonda Ball      RB  18  running back   young        5        18
6   6       Hugo Long     WR  23 wide receiver   young        6        23
7   7 Lionel Scrimmage     WR  27 wide receiver   young        7        27
8   8      Drew Blood     WR  32 wide receiver     old        8        32
9   9    Chase Emdown     WR  26 wide receiver   young        9        26
10 10     Justin Time      TE  23    tight end   young       10        23
11 11    Spike D'Ball      TE  NA    tight end    <NA>       11        NA
12 12      Isac Ulooz      LB  37         <NA>     old       12        37
   integerVar characterVar fantasyPoints
1          NA          old            NA
2          NA          old           250
3          NA        young            NA
4          NA        young            NA
5          NA        young            NA
6          NA        young            NA
7          NA        young           170
8          NA          old            NA
9          NA        young            NA
10         NA        young            NA
11         NA         <NA>            NA
12         NA          old            NA
```

```
dim(leftJoinData)
```

```
[1] 12 11
```

3.21.3.4 Right Outer Join

A right outer join includes all rows in `y`. It returns columns from `x` and `y`. Here is how to merge two data frames using a right outer join ("right join"):

```
rightJoinData <- right_join(
  players,
  fantasyPoints,
  by = "ID")

rightJoinData
```

```
  ID          name position age      newVar1 newVar2 factorVar numericVar
1  2      Ben Sacked      QB  30   quarterback     old        2        30
2  7 Lionel Scrimmage      WR  27 wide receiver   young        7        27
3 13            <NA>     <NA>  NA         <NA>    <NA>      <NA>        NA
4 14            <NA>     <NA>  NA         <NA>    <NA>      <NA>        NA
  integerVar characterVar fantasyPoints
1         NA          old           250
2         NA        young           170
3         NA         <NA>            65
4         NA         <NA>            15
```

```
dim(rightJoinData)
```

```
[1]  4 11
```

3.21.3.5 Inner Join

An inner join includes all rows that are in **both** `x` **and** `y`. An inner join will return one row of `x` for each matching row of `y`, and can duplicate values of records on either side (left or right) if `x` and `y` have more than one matching record. It returns columns from `x` and `y`. Here is how to merge two data frames using an inner join:

```
innerJoinData <- inner_join(
  players,
  fantasyPoints,
  by = "ID")

innerJoinData
```

```
  ID          name position age      newVar1 newVar2 factorVar numericVar
```

```
1  2      Ben Sacked      QB  30   quarterback    old        2        30
2  7 Lionel Scrimmage      WR  27 wide receiver   young       7        27
  integerVar characterVar fantasyPoints
1         NA          old           250
2         NA        young           170
```

```
dim(innerJoinData)
```

```
[1]  2 11
```

3.21.3.6 Semi Join

A semi join is a filter. A left semi join returns all rows from `x` **with** a match in `y`. That is, it filters out records from `x` that are not in `y`. Unlike an inner join, a left semi join will never duplicate rows of `x`, and it includes columns from only `x` (not from `y`). Here is how to merge two data frames using a left semi join:

```
semiJoinData <- semi_join(
  players,
  fantasyPoints,
  by = "ID")

semiJoinData
```

```
  ID          name position age      newVar1 newVar2 factorVar numericVar
1  2      Ben Sacked      QB  30   quarterback    old        2        30
2  7 Lionel Scrimmage      WR  27 wide receiver   young       7        27
  integerVar characterVar
1         NA          old
2         NA        young
```

```
dim(semiJoinData)
```

```
[1]  2 10
```

3.21.3.7 Anti Join

An anti join is a filter. A left anti join returns all rows from `x` **without** a match in `y`. That is, it filters out records from `x` that are in `y`. It returns columns from only `x` (not from `y`). Here is how to merge two data frames using a left anti join:

```
antiJoinData <- anti_join(
  players,
  fantasyPoints,
  by = "ID")

antiJoinData
```

```
   ID         name position age      newVar1 newVar2 factorVar numericVar
1   1    Ken Cussion      QB  40   quarterback     old        1        40
2   3 Chuck Downfield      QB  24   quarterback   young        3        24
3   4    Ron Ingback      RB  20  running back   young        4        20
4   5    Rhonda Ball      RB  18  running back   young        5        18
5   6      Hugo Long      WR  23 wide receiver   young        6        23
6   8     Drew Blood      WR  32 wide receiver     old        8        32
7   9   Chase Emdown      WR  26 wide receiver   young        9        26
8  10    Justin Time      TE  23     tight end   young       10        23
9  11   Spike D'Ball      TE  NA     tight end    <NA>       11        NA
10 12     Isac Ulooz      LB  37         <NA>     old       12        37
   integerVar characterVar
1          NA          old
2          NA        young
3          NA        young
4          NA        young
5          NA        young
6          NA          old
7          NA        young
8          NA        young
9          NA         <NA>
10         NA          old
```

```
dim(antiJoinData)
```

```
[1] 10 10
```

3.21.3.8 Cross Join

A cross join combines each row in `x` with each row in `y`.

```
crossJoinData <- cross_join(
  players,
  fantasyPoints)

crossJoinData
```

	ID.x	name	position	age	newVar1	newVar2	factorVar
1	1	Ken Cussion	QB	40	quarterback	old	1
2	1	Ken Cussion	QB	40	quarterback	old	1
3	1	Ken Cussion	QB	40	quarterback	old	1
4	1	Ken Cussion	QB	40	quarterback	old	1
5	2	Ben Sacked	QB	30	quarterback	old	2
6	2	Ben Sacked	QB	30	quarterback	old	2
7	2	Ben Sacked	QB	30	quarterback	old	2
8	2	Ben Sacked	QB	30	quarterback	old	2
9	3	Chuck Downfield	QB	24	quarterback	young	3
10	3	Chuck Downfield	QB	24	quarterback	young	3
11	3	Chuck Downfield	QB	24	quarterback	young	3
12	3	Chuck Downfield	QB	24	quarterback	young	3
13	4	Ron Ingback	RB	20	running back	young	4
14	4	Ron Ingback	RB	20	running back	young	4
15	4	Ron Ingback	RB	20	running back	young	4
16	4	Ron Ingback	RB	20	running back	young	4
17	5	Rhonda Ball	RB	18	running back	young	5
18	5	Rhonda Ball	RB	18	running back	young	5
19	5	Rhonda Ball	RB	18	running back	young	5
20	5	Rhonda Ball	RB	18	running back	young	5
21	6	Hugo Long	WR	23	wide receiver	young	6
22	6	Hugo Long	WR	23	wide receiver	young	6
23	6	Hugo Long	WR	23	wide receiver	young	6
24	6	Hugo Long	WR	23	wide receiver	young	6
25	7	Lionel Scrimmage	WR	27	wide receiver	young	7
26	7	Lionel Scrimmage	WR	27	wide receiver	young	7
27	7	Lionel Scrimmage	WR	27	wide receiver	young	7
28	7	Lionel Scrimmage	WR	27	wide receiver	young	7
29	8	Drew Blood	WR	32	wide receiver	old	8
30	8	Drew Blood	WR	32	wide receiver	old	8
31	8	Drew Blood	WR	32	wide receiver	old	8
32	8	Drew Blood	WR	32	wide receiver	old	8
33	9	Chase Emdown	WR	26	wide receiver	young	9
34	9	Chase Emdown	WR	26	wide receiver	young	9
35	9	Chase Emdown	WR	26	wide receiver	young	9
36	9	Chase Emdown	WR	26	wide receiver	young	9
37	10	Justin Time	TE	23	tight end	young	10
38	10	Justin Time	TE	23	tight end	young	10
39	10	Justin Time	TE	23	tight end	young	10
40	10	Justin Time	TE	23	tight end	young	10
41	11	Spike D'Ball	TE	NA	tight end	<NA>	11
42	11	Spike D'Ball	TE	NA	tight end	<NA>	11
43	11	Spike D'Ball	TE	NA	tight end	<NA>	11
44	11	Spike D'Ball	TE	NA	tight end	<NA>	11

```
45   12       Isac Ulooz       LB  37          <NA>     old        12
46   12       Isac Ulooz       LB  37          <NA>     old        12
47   12       Isac Ulooz       LB  37          <NA>     old        12
48   12       Isac Ulooz       LB  37          <NA>     old        12
   numericVar integerVar characterVar ID.y fantasyPoints
1          40         NA          old    2           250
2          40         NA          old    7           170
3          40         NA          old   13            65
4          40         NA          old   14            15
5          30         NA          old    2           250
6          30         NA          old    7           170
7          30         NA          old   13            65
8          30         NA          old   14            15
9          24         NA        young    2           250
10         24         NA        young    7           170
11         24         NA        young   13            65
12         24         NA        young   14            15
13         20         NA        young    2           250
14         20         NA        young    7           170
15         20         NA        young   13            65
16         20         NA        young   14            15
17         18         NA        young    2           250
18         18         NA        young    7           170
19         18         NA        young   13            65
20         18         NA        young   14            15
21         23         NA        young    2           250
22         23         NA        young    7           170
23         23         NA        young   13            65
24         23         NA        young   14            15
25         27         NA        young    2           250
26         27         NA        young    7           170
27         27         NA        young   13            65
28         27         NA        young   14            15
29         32         NA          old    2           250
30         32         NA          old    7           170
31         32         NA          old   13            65
32         32         NA          old   14            15
33         26         NA        young    2           250
34         26         NA        young    7           170
35         26         NA        young   13            65
36         26         NA        young   14            15
37         23         NA        young    2           250
38         23         NA        young    7           170
39         23         NA        young   13            65
40         23         NA        young   14            15
```

```
41          NA          NA          <NA>    2           250
42          NA          NA          <NA>    7           170
43          NA          NA          <NA>   13            65
44          NA          NA          <NA>   14            15
45          37          NA           old    2           250
46          37          NA           old    7           170
47          37          NA           old   13            65
48          37          NA           old   14            15
```

```
dim(crossJoinData)
```

```
[1] 48 12
```

3.22 Transform Data from Long to Wide

Depending on the analysis, it may be important to restructure the data to be in long or wide form. When the data are in wide form, each player has only one row. When the data are in long form, each player has multiple rows—e.g., a row for each game. The data structure is called wide or long form because a dataset in wide form has more columns and fewer rows (i.e., it appears wider and shorter), whereas a dataset in long form has more rows and fewer columns (i.e., it appears narrower and taller).

Here are the original data in long form. The data are structured in "`player-season-week` form". That is, every row in the dataset is uniquely identified by the combination of variables, `ID`, `season`, and `week`—these are the keys. This is an example of long form, because each player has multiple rows.

```
dataLong <- full_join(
  players %>% select(-age),
  fantasyPoints_weekly,
  by = c("ID")
)

dataLong
```

```
   ID           name position      newVar1 newVar2 factorVar numericVar
1   1      Ken Cussion       QB   quarterback     old        1        40
2   1      Ken Cussion       QB   quarterback     old        1        40
3   1      Ken Cussion       QB   quarterback     old        1        40
4   1      Ken Cussion       QB   quarterback     old        1        40
```

5	1	Ken Cussion	QB	quarterback	old	1	40
6	1	Ken Cussion	QB	quarterback	old	1	40
7	1	Ken Cussion	QB	quarterback	old	1	40
8	1	Ken Cussion	QB	quarterback	old	1	40
9	1	Ken Cussion	QB	quarterback	old	1	40
10	1	Ken Cussion	QB	quarterback	old	1	40
11	1	Ken Cussion	QB	quarterback	old	1	40
12	1	Ken Cussion	QB	quarterback	old	1	40
13	1	Ken Cussion	QB	quarterback	old	1	40
14	1	Ken Cussion	QB	quarterback	old	1	40
15	1	Ken Cussion	QB	quarterback	old	1	40
16	1	Ken Cussion	QB	quarterback	old	1	40
17	1	Ken Cussion	QB	quarterback	old	1	40
18	1	Ken Cussion	QB	quarterback	old	1	40
19	1	Ken Cussion	QB	quarterback	old	1	40
20	1	Ken Cussion	QB	quarterback	old	1	40
21	1	Ken Cussion	QB	quarterback	old	1	40
22	1	Ken Cussion	QB	quarterback	old	1	40
23	1	Ken Cussion	QB	quarterback	old	1	40
24	1	Ken Cussion	QB	quarterback	old	1	40
25	1	Ken Cussion	QB	quarterback	old	1	40
26	1	Ken Cussion	QB	quarterback	old	1	40
27	1	Ken Cussion	QB	quarterback	old	1	40
28	1	Ken Cussion	QB	quarterback	old	1	40
29	1	Ken Cussion	QB	quarterback	old	1	40
30	1	Ken Cussion	QB	quarterback	old	1	40
31	1	Ken Cussion	QB	quarterback	old	1	40
32	1	Ken Cussion	QB	quarterback	old	1	40
33	1	Ken Cussion	QB	quarterback	old	1	40
34	1	Ken Cussion	QB	quarterback	old	1	40
35	2	Ben Sacked	QB	quarterback	old	2	30
36	2	Ben Sacked	QB	quarterback	old	2	30
37	2	Ben Sacked	QB	quarterback	old	2	30
38	2	Ben Sacked	QB	quarterback	old	2	30
39	2	Ben Sacked	QB	quarterback	old	2	30
40	2	Ben Sacked	QB	quarterback	old	2	30
41	2	Ben Sacked	QB	quarterback	old	2	30
42	2	Ben Sacked	QB	quarterback	old	2	30
43	2	Ben Sacked	QB	quarterback	old	2	30
44	2	Ben Sacked	QB	quarterback	old	2	30
45	2	Ben Sacked	QB	quarterback	old	2	30
46	2	Ben Sacked	QB	quarterback	old	2	30
47	2	Ben Sacked	QB	quarterback	old	2	30
48	2	Ben Sacked	QB	quarterback	old	2	30
49	2	Ben Sacked	QB	quarterback	old	2	30

50	2	Ben Sacked	QB	quarterback	old	2	30
51	2	Ben Sacked	QB	quarterback	old	2	30
52	2	Ben Sacked	QB	quarterback	old	2	30
53	2	Ben Sacked	QB	quarterback	old	2	30
54	2	Ben Sacked	QB	quarterback	old	2	30
55	2	Ben Sacked	QB	quarterback	old	2	30
56	2	Ben Sacked	QB	quarterback	old	2	30
57	2	Ben Sacked	QB	quarterback	old	2	30
58	2	Ben Sacked	QB	quarterback	old	2	30
59	2	Ben Sacked	QB	quarterback	old	2	30
60	2	Ben Sacked	QB	quarterback	old	2	30
61	2	Ben Sacked	QB	quarterback	old	2	30
62	2	Ben Sacked	QB	quarterback	old	2	30
63	2	Ben Sacked	QB	quarterback	old	2	30
64	2	Ben Sacked	QB	quarterback	old	2	30
65	2	Ben Sacked	QB	quarterback	old	2	30
66	2	Ben Sacked	QB	quarterback	old	2	30
67	2	Ben Sacked	QB	quarterback	old	2	30
68	2	Ben Sacked	QB	quarterback	old	2	30
69	3	Chuck Downfield	QB	quarterback	young	3	24
70	3	Chuck Downfield	QB	quarterback	young	3	24
71	3	Chuck Downfield	QB	quarterback	young	3	24
72	3	Chuck Downfield	QB	quarterback	young	3	24
73	3	Chuck Downfield	QB	quarterback	young	3	24
74	3	Chuck Downfield	QB	quarterback	young	3	24
75	3	Chuck Downfield	QB	quarterback	young	3	24
76	3	Chuck Downfield	QB	quarterback	young	3	24
77	3	Chuck Downfield	QB	quarterback	young	3	24
78	3	Chuck Downfield	QB	quarterback	young	3	24
79	3	Chuck Downfield	QB	quarterback	young	3	24
80	3	Chuck Downfield	QB	quarterback	young	3	24
81	3	Chuck Downfield	QB	quarterback	young	3	24
82	3	Chuck Downfield	QB	quarterback	young	3	24
83	3	Chuck Downfield	QB	quarterback	young	3	24
84	3	Chuck Downfield	QB	quarterback	young	3	24
85	3	Chuck Downfield	QB	quarterback	young	3	24
86	3	Chuck Downfield	QB	quarterback	young	3	24
87	3	Chuck Downfield	QB	quarterback	young	3	24
88	3	Chuck Downfield	QB	quarterback	young	3	24
89	3	Chuck Downfield	QB	quarterback	young	3	24
90	3	Chuck Downfield	QB	quarterback	young	3	24
91	3	Chuck Downfield	QB	quarterback	young	3	24
92	3	Chuck Downfield	QB	quarterback	young	3	24
93	3	Chuck Downfield	QB	quarterback	young	3	24
94	3	Chuck Downfield	QB	quarterback	young	3	24

```
95   3  Chuck Downfield      QB   quarterback   young        3        24
96   3  Chuck Downfield      QB   quarterback   young        3        24
97   3  Chuck Downfield      QB   quarterback   young        3        24
98   3  Chuck Downfield      QB   quarterback   young        3        24
99   3  Chuck Downfield      QB   quarterback   young        3        24
100  3  Chuck Downfield      QB   quarterback   young        3        24
101  3  Chuck Downfield      QB   quarterback   young        3        24
102  3  Chuck Downfield      QB   quarterback   young        3        24
103  4      Ron Ingback      RB  running back   young        4        20
104  4      Ron Ingback      RB  running back   young        4        20
105  4      Ron Ingback      RB  running back   young        4        20
106  4      Ron Ingback      RB  running back   young        4        20
107  4      Ron Ingback      RB  running back   young        4        20
108  4      Ron Ingback      RB  running back   young        4        20
109  4      Ron Ingback      RB  running back   young        4        20
110  4      Ron Ingback      RB  running back   young        4        20
111  4      Ron Ingback      RB  running back   young        4        20
112  4      Ron Ingback      RB  running back   young        4        20
113  4      Ron Ingback      RB  running back   young        4        20
114  4      Ron Ingback      RB  running back   young        4        20
115  4      Ron Ingback      RB  running back   young        4        20
116  4      Ron Ingback      RB  running back   young        4        20
117  4      Ron Ingback      RB  running back   young        4        20
118  4      Ron Ingback      RB  running back   young        4        20
119  4      Ron Ingback      RB  running back   young        4        20
120  4      Ron Ingback      RB  running back   young        4        20
121  4      Ron Ingback      RB  running back   young        4        20
122  4      Ron Ingback      RB  running back   young        4        20
123  4      Ron Ingback      RB  running back   young        4        20
124  4      Ron Ingback      RB  running back   young        4        20
125  4      Ron Ingback      RB  running back   young        4        20
126  4      Ron Ingback      RB  running back   young        4        20
127  4      Ron Ingback      RB  running back   young        4        20
128  4      Ron Ingback      RB  running back   young        4        20
129  4      Ron Ingback      RB  running back   young        4        20
130  4      Ron Ingback      RB  running back   young        4        20
131  4      Ron Ingback      RB  running back   young        4        20
132  4      Ron Ingback      RB  running back   young        4        20
133  4      Ron Ingback      RB  running back   young        4        20
134  4      Ron Ingback      RB  running back   young        4        20
135  4      Ron Ingback      RB  running back   young        4        20
136  4      Ron Ingback      RB  running back   young        4        20
137  5      Rhonda Ball      RB  running back   young        5        18
138  5      Rhonda Ball      RB  running back   young        5        18
139  5      Rhonda Ball      RB  running back   young        5        18
```

140	5	Rhonda Ball	RB	running back	young	5	18
141	5	Rhonda Ball	RB	running back	young	5	18
142	5	Rhonda Ball	RB	running back	young	5	18
143	5	Rhonda Ball	RB	running back	young	5	18
144	5	Rhonda Ball	RB	running back	young	5	18
145	5	Rhonda Ball	RB	running back	young	5	18
146	5	Rhonda Ball	RB	running back	young	5	18
147	5	Rhonda Ball	RB	running back	young	5	18
148	5	Rhonda Ball	RB	running back	young	5	18
149	5	Rhonda Ball	RB	running back	young	5	18
150	5	Rhonda Ball	RB	running back	young	5	18
151	5	Rhonda Ball	RB	running back	young	5	18
152	5	Rhonda Ball	RB	running back	young	5	18
153	5	Rhonda Ball	RB	running back	young	5	18
154	5	Rhonda Ball	RB	running back	young	5	18
155	5	Rhonda Ball	RB	running back	young	5	18
156	5	Rhonda Ball	RB	running back	young	5	18
157	5	Rhonda Ball	RB	running back	young	5	18
158	5	Rhonda Ball	RB	running back	young	5	18
159	5	Rhonda Ball	RB	running back	young	5	18
160	5	Rhonda Ball	RB	running back	young	5	18
161	5	Rhonda Ball	RB	running back	young	5	18
162	5	Rhonda Ball	RB	running back	young	5	18
163	5	Rhonda Ball	RB	running back	young	5	18
164	5	Rhonda Ball	RB	running back	young	5	18
165	5	Rhonda Ball	RB	running back	young	5	18
166	5	Rhonda Ball	RB	running back	young	5	18
167	5	Rhonda Ball	RB	running back	young	5	18
168	5	Rhonda Ball	RB	running back	young	5	18
169	5	Rhonda Ball	RB	running back	young	5	18
170	5	Rhonda Ball	RB	running back	young	5	18
171	6	Hugo Long	WR	wide receiver	young	6	23
172	6	Hugo Long	WR	wide receiver	young	6	23
173	6	Hugo Long	WR	wide receiver	young	6	23
174	6	Hugo Long	WR	wide receiver	young	6	23
175	6	Hugo Long	WR	wide receiver	young	6	23
176	6	Hugo Long	WR	wide receiver	young	6	23
177	6	Hugo Long	WR	wide receiver	young	6	23
178	6	Hugo Long	WR	wide receiver	young	6	23
179	6	Hugo Long	WR	wide receiver	young	6	23
180	6	Hugo Long	WR	wide receiver	young	6	23
181	6	Hugo Long	WR	wide receiver	young	6	23
182	6	Hugo Long	WR	wide receiver	young	6	23
183	6	Hugo Long	WR	wide receiver	young	6	23
184	6	Hugo Long	WR	wide receiver	young	6	23

```
185  6        Hugo Long      WR wide receiver   young        6        23
186  6        Hugo Long      WR wide receiver   young        6        23
187  6        Hugo Long      WR wide receiver   young        6        23
188  6        Hugo Long      WR wide receiver   young        6        23
189  6        Hugo Long      WR wide receiver   young        6        23
190  6        Hugo Long      WR wide receiver   young        6        23
191  6        Hugo Long      WR wide receiver   young        6        23
192  6        Hugo Long      WR wide receiver   young        6        23
193  6        Hugo Long      WR wide receiver   young        6        23
194  6        Hugo Long      WR wide receiver   young        6        23
195  6        Hugo Long      WR wide receiver   young        6        23
196  6        Hugo Long      WR wide receiver   young        6        23
197  6        Hugo Long      WR wide receiver   young        6        23
198  6        Hugo Long      WR wide receiver   young        6        23
199  6        Hugo Long      WR wide receiver   young        6        23
200  6        Hugo Long      WR wide receiver   young        6        23
201  6        Hugo Long      WR wide receiver   young        6        23
202  6        Hugo Long      WR wide receiver   young        6        23
203  6        Hugo Long      WR wide receiver   young        6        23
204  6        Hugo Long      WR wide receiver   young        6        23
205  7 Lionel Scrimmage      WR wide receiver   young        7        27
206  7 Lionel Scrimmage      WR wide receiver   young        7        27
207  7 Lionel Scrimmage      WR wide receiver   young        7        27
208  7 Lionel Scrimmage      WR wide receiver   young        7        27
209  7 Lionel Scrimmage      WR wide receiver   young        7        27
210  7 Lionel Scrimmage      WR wide receiver   young        7        27
211  7 Lionel Scrimmage      WR wide receiver   young        7        27
212  7 Lionel Scrimmage      WR wide receiver   young        7        27
213  7 Lionel Scrimmage      WR wide receiver   young        7        27
214  7 Lionel Scrimmage      WR wide receiver   young        7        27
215  7 Lionel Scrimmage      WR wide receiver   young        7        27
216  7 Lionel Scrimmage      WR wide receiver   young        7        27
217  7 Lionel Scrimmage      WR wide receiver   young        7        27
218  7 Lionel Scrimmage      WR wide receiver   young        7        27
219  7 Lionel Scrimmage      WR wide receiver   young        7        27
220  7 Lionel Scrimmage      WR wide receiver   young        7        27
221  7 Lionel Scrimmage      WR wide receiver   young        7        27
222  7 Lionel Scrimmage      WR wide receiver   young        7        27
223  7 Lionel Scrimmage      WR wide receiver   young        7        27
224  7 Lionel Scrimmage      WR wide receiver   young        7        27
225  7 Lionel Scrimmage      WR wide receiver   young        7        27
226  7 Lionel Scrimmage      WR wide receiver   young        7        27
227  7 Lionel Scrimmage      WR wide receiver   young        7        27
228  7 Lionel Scrimmage      WR wide receiver   young        7        27
229  7 Lionel Scrimmage      WR wide receiver   young        7        27
```

230	7	Lionel Scrimmage	WR wide receiver	young	7	27
231	7	Lionel Scrimmage	WR wide receiver	young	7	27
232	7	Lionel Scrimmage	WR wide receiver	young	7	27
233	7	Lionel Scrimmage	WR wide receiver	young	7	27
234	7	Lionel Scrimmage	WR wide receiver	young	7	27
235	7	Lionel Scrimmage	WR wide receiver	young	7	27
236	7	Lionel Scrimmage	WR wide receiver	young	7	27
237	7	Lionel Scrimmage	WR wide receiver	young	7	27
238	7	Lionel Scrimmage	WR wide receiver	young	7	27
239	8	Drew Blood	WR wide receiver	old	8	32
240	8	Drew Blood	WR wide receiver	old	8	32
241	8	Drew Blood	WR wide receiver	old	8	32
242	8	Drew Blood	WR wide receiver	old	8	32
243	8	Drew Blood	WR wide receiver	old	8	32
244	8	Drew Blood	WR wide receiver	old	8	32
245	8	Drew Blood	WR wide receiver	old	8	32
246	8	Drew Blood	WR wide receiver	old	8	32
247	8	Drew Blood	WR wide receiver	old	8	32
248	8	Drew Blood	WR wide receiver	old	8	32
249	8	Drew Blood	WR wide receiver	old	8	32
250	8	Drew Blood	WR wide receiver	old	8	32
251	8	Drew Blood	WR wide receiver	old	8	32
252	8	Drew Blood	WR wide receiver	old	8	32
253	8	Drew Blood	WR wide receiver	old	8	32
254	8	Drew Blood	WR wide receiver	old	8	32
255	8	Drew Blood	WR wide receiver	old	8	32
256	8	Drew Blood	WR wide receiver	old	8	32
257	8	Drew Blood	WR wide receiver	old	8	32
258	8	Drew Blood	WR wide receiver	old	8	32
259	8	Drew Blood	WR wide receiver	old	8	32
260	8	Drew Blood	WR wide receiver	old	8	32
261	8	Drew Blood	WR wide receiver	old	8	32
262	8	Drew Blood	WR wide receiver	old	8	32
263	8	Drew Blood	WR wide receiver	old	8	32
264	8	Drew Blood	WR wide receiver	old	8	32
265	8	Drew Blood	WR wide receiver	old	8	32
266	8	Drew Blood	WR wide receiver	old	8	32
267	8	Drew Blood	WR wide receiver	old	8	32
268	8	Drew Blood	WR wide receiver	old	8	32
269	8	Drew Blood	WR wide receiver	old	8	32
270	8	Drew Blood	WR wide receiver	old	8	32
271	8	Drew Blood	WR wide receiver	old	8	32
272	8	Drew Blood	WR wide receiver	old	8	32
273	9	Chase Emdown	WR wide receiver	young	9	26
274	9	Chase Emdown	WR wide receiver	young	9	26

```
275  9    Chase Emdown      WR wide receiver   young        9        26
276  9    Chase Emdown      WR wide receiver   young        9        26
277  9    Chase Emdown      WR wide receiver   young        9        26
278  9    Chase Emdown      WR wide receiver   young        9        26
279  9    Chase Emdown      WR wide receiver   young        9        26
280  9    Chase Emdown      WR wide receiver   young        9        26
281  9    Chase Emdown      WR wide receiver   young        9        26
282  9    Chase Emdown      WR wide receiver   young        9        26
283  9    Chase Emdown      WR wide receiver   young        9        26
284  9    Chase Emdown      WR wide receiver   young        9        26
285  9    Chase Emdown      WR wide receiver   young        9        26
286  9    Chase Emdown      WR wide receiver   young        9        26
287  9    Chase Emdown      WR wide receiver   young        9        26
288  9    Chase Emdown      WR wide receiver   young        9        26
289  9    Chase Emdown      WR wide receiver   young        9        26
290  9    Chase Emdown      WR wide receiver   young        9        26
291  9    Chase Emdown      WR wide receiver   young        9        26
292  9    Chase Emdown      WR wide receiver   young        9        26
293  9    Chase Emdown      WR wide receiver   young        9        26
294  9    Chase Emdown      WR wide receiver   young        9        26
295  9    Chase Emdown      WR wide receiver   young        9        26
296  9    Chase Emdown      WR wide receiver   young        9        26
297  9    Chase Emdown      WR wide receiver   young        9        26
298  9    Chase Emdown      WR wide receiver   young        9        26
299  9    Chase Emdown      WR wide receiver   young        9        26
300  9    Chase Emdown      WR wide receiver   young        9        26
301  9    Chase Emdown      WR wide receiver   young        9        26
302  9    Chase Emdown      WR wide receiver   young        9        26
303  9    Chase Emdown      WR wide receiver   young        9        26
304  9    Chase Emdown      WR wide receiver   young        9        26
305  9    Chase Emdown      WR wide receiver   young        9        26
306  9    Chase Emdown      WR wide receiver   young        9        26
307 10     Justin Time      TE     tight end   young       10        23
308 10     Justin Time      TE     tight end   young       10        23
309 10     Justin Time      TE     tight end   young       10        23
310 10     Justin Time      TE     tight end   young       10        23
311 10     Justin Time      TE     tight end   young       10        23
312 10     Justin Time      TE     tight end   young       10        23
313 10     Justin Time      TE     tight end   young       10        23
314 10     Justin Time      TE     tight end   young       10        23
315 10     Justin Time      TE     tight end   young       10        23
316 10     Justin Time      TE     tight end   young       10        23
317 10     Justin Time      TE     tight end   young       10        23
318 10     Justin Time      TE     tight end   young       10        23
319 10     Justin Time      TE     tight end   young       10        23
```

```
320 10     Justin Time      TE    tight end   young       10        23
321 10     Justin Time      TE    tight end   young       10        23
322 10     Justin Time      TE    tight end   young       10        23
323 10     Justin Time      TE    tight end   young       10        23
324 10     Justin Time      TE    tight end   young       10        23
325 10     Justin Time      TE    tight end   young       10        23
326 10     Justin Time      TE    tight end   young       10        23
327 10     Justin Time      TE    tight end   young       10        23
328 10     Justin Time      TE    tight end   young       10        23
329 10     Justin Time      TE    tight end   young       10        23
330 10     Justin Time      TE    tight end   young       10        23
331 10     Justin Time      TE    tight end   young       10        23
332 10     Justin Time      TE    tight end   young       10        23
333 10     Justin Time      TE    tight end   young       10        23
334 10     Justin Time      TE    tight end   young       10        23
335 10     Justin Time      TE    tight end   young       10        23
336 10     Justin Time      TE    tight end   young       10        23
337 10     Justin Time      TE    tight end   young       10        23
338 10     Justin Time      TE    tight end   young       10        23
339 10     Justin Time      TE    tight end   young       10        23
340 10     Justin Time      TE    tight end   young       10        23
341 11    Spike D'Ball      TE    tight end    <NA>       11        NA
342 11    Spike D'Ball      TE    tight end    <NA>       11        NA
343 11    Spike D'Ball      TE    tight end    <NA>       11        NA
344 11    Spike D'Ball      TE    tight end    <NA>       11        NA
345 11    Spike D'Ball      TE    tight end    <NA>       11        NA
346 11    Spike D'Ball      TE    tight end    <NA>       11        NA
347 11    Spike D'Ball      TE    tight end    <NA>       11        NA
348 11    Spike D'Ball      TE    tight end    <NA>       11        NA
349 11    Spike D'Ball      TE    tight end    <NA>       11        NA
350 11    Spike D'Ball      TE    tight end    <NA>       11        NA
351 11    Spike D'Ball      TE    tight end    <NA>       11        NA
352 11    Spike D'Ball      TE    tight end    <NA>       11        NA
353 11    Spike D'Ball      TE    tight end    <NA>       11        NA
354 11    Spike D'Ball      TE    tight end    <NA>       11        NA
355 11    Spike D'Ball      TE    tight end    <NA>       11        NA
356 11    Spike D'Ball      TE    tight end    <NA>       11        NA
357 11    Spike D'Ball      TE    tight end    <NA>       11        NA
358 11    Spike D'Ball      TE    tight end    <NA>       11        NA
359 11    Spike D'Ball      TE    tight end    <NA>       11        NA
360 11    Spike D'Ball      TE    tight end    <NA>       11        NA
361 11    Spike D'Ball      TE    tight end    <NA>       11        NA
362 11    Spike D'Ball      TE    tight end    <NA>       11        NA
363 11    Spike D'Ball      TE    tight end    <NA>       11        NA
364 11    Spike D'Ball      TE    tight end    <NA>       11        NA
```

```
365 11    Spike D'Ball      TE    tight end   <NA>       11        NA
366 11    Spike D'Ball      TE    tight end   <NA>       11        NA
367 11    Spike D'Ball      TE    tight end   <NA>       11        NA
368 11    Spike D'Ball      TE    tight end   <NA>       11        NA
369 11    Spike D'Ball      TE    tight end   <NA>       11        NA
370 11    Spike D'Ball      TE    tight end   <NA>       11        NA
371 11    Spike D'Ball      TE    tight end   <NA>       11        NA
372 11    Spike D'Ball      TE    tight end   <NA>       11        NA
373 11    Spike D'Ball      TE    tight end   <NA>       11        NA
374 11    Spike D'Ball      TE    tight end   <NA>       11        NA
375 12      Isac Ulooz      LB         <NA>    old       12        37
376 12      Isac Ulooz      LB         <NA>    old       12        37
377 12      Isac Ulooz      LB         <NA>    old       12        37
378 12      Isac Ulooz      LB         <NA>    old       12        37
379 12      Isac Ulooz      LB         <NA>    old       12        37
380 12      Isac Ulooz      LB         <NA>    old       12        37
381 12      Isac Ulooz      LB         <NA>    old       12        37
382 12      Isac Ulooz      LB         <NA>    old       12        37
383 12      Isac Ulooz      LB         <NA>    old       12        37
384 12      Isac Ulooz      LB         <NA>    old       12        37
385 12      Isac Ulooz      LB         <NA>    old       12        37
386 12      Isac Ulooz      LB         <NA>    old       12        37
387 12      Isac Ulooz      LB         <NA>    old       12        37
388 12      Isac Ulooz      LB         <NA>    old       12        37
389 12      Isac Ulooz      LB         <NA>    old       12        37
390 12      Isac Ulooz      LB         <NA>    old       12        37
391 12      Isac Ulooz      LB         <NA>    old       12        37
392 12      Isac Ulooz      LB         <NA>    old       12        37
393 12      Isac Ulooz      LB         <NA>    old       12        37
394 12      Isac Ulooz      LB         <NA>    old       12        37
395 12      Isac Ulooz      LB         <NA>    old       12        37
396 12      Isac Ulooz      LB         <NA>    old       12        37
397 12      Isac Ulooz      LB         <NA>    old       12        37
398 12      Isac Ulooz      LB         <NA>    old       12        37
399 12      Isac Ulooz      LB         <NA>    old       12        37
400 12      Isac Ulooz      LB         <NA>    old       12        37
401 12      Isac Ulooz      LB         <NA>    old       12        37
402 12      Isac Ulooz      LB         <NA>    old       12        37
403 12      Isac Ulooz      LB         <NA>    old       12        37
404 12      Isac Ulooz      LB         <NA>    old       12        37
405 12      Isac Ulooz      LB         <NA>    old       12        37
406 12      Isac Ulooz      LB         <NA>    old       12        37
407 12      Isac Ulooz      LB         <NA>    old       12        37
408 12      Isac Ulooz      LB         <NA>    old       12        37
    integerVar characterVar season week fantasyPoints
```

1	NA	old	2022	1	24
2	NA	old	2023	1	4
3	NA	old	2022	2	30
4	NA	old	2023	2	4
5	NA	old	2022	3	27
6	NA	old	2023	3	3
7	NA	old	2022	4	5
8	NA	old	2023	4	22
9	NA	old	2022	5	14
10	NA	old	2023	5	31
11	NA	old	2022	6	23
12	NA	old	2023	6	23
13	NA	old	2022	7	12
14	NA	old	2023	7	32
15	NA	old	2022	8	35
16	NA	old	2023	8	34
17	NA	old	2022	9	6
18	NA	old	2023	9	4
19	NA	old	2022	10	4
20	NA	old	2023	10	8
21	NA	old	2022	11	13
22	NA	old	2023	11	23
23	NA	old	2022	12	4
24	NA	old	2023	12	9
25	NA	old	2022	13	30
26	NA	old	2023	13	18
27	NA	old	2022	14	11
28	NA	old	2023	14	31
29	NA	old	2022	15	9
30	NA	old	2023	15	19
31	NA	old	2022	16	4
32	NA	old	2023	16	7
33	NA	old	2022	17	30
34	NA	old	2023	17	5
35	NA	old	2022	1	32
36	NA	old	2023	1	21
37	NA	old	2022	2	35
38	NA	old	2023	2	9
39	NA	old	2022	3	6
40	NA	old	2023	3	3
41	NA	old	2022	4	13
42	NA	old	2023	4	35
43	NA	old	2022	5	0
44	NA	old	2023	5	31
45	NA	old	2022	6	28

```
46          NA          old   2023    6             0
47          NA          old   2022    7             6
48          NA          old   2023    7            23
49          NA          old   2022    8            19
50          NA          old   2023    8            33
51          NA          old   2022    9             9
52          NA          old   2023    9             7
53          NA          old   2022   10            20
54          NA          old   2023   10             4
55          NA          old   2022   11            14
56          NA          old   2023   11            21
57          NA          old   2022   12            35
58          NA          old   2023   12            26
59          NA          old   2022   13            27
60          NA          old   2023   13            16
61          NA          old   2022   14             9
62          NA          old   2023   14             2
63          NA          old   2022   15            30
64          NA          old   2023   15             6
65          NA          old   2022   16             6
66          NA          old   2023   16             0
67          NA          old   2022   17             5
68          NA          old   2023   17             0
69          NA        young   2022    1             7
70          NA        young   2023    1            24
71          NA        young   2022    2             0
72          NA        young   2023    2            31
73          NA        young   2022    3            10
74          NA        young   2023    3             2
75          NA        young   2022    4            18
76          NA        young   2023    4            12
77          NA        young   2022    5            28
78          NA        young   2023    5            22
79          NA        young   2022    6             5
80          NA        young   2023    6            12
81          NA        young   2022    7             0
82          NA        young   2023    7            14
83          NA        young   2022    8            17
84          NA        young   2023    8            20
85          NA        young   2022    9            17
86          NA        young   2023    9            29
87          NA        young   2022   10            30
88          NA        young   2023   10            11
89          NA        young   2022   11            24
90          NA        young   2023   11            11
```

91	NA	young	2022	12	4
92	NA	young	2023	12	30
93	NA	young	2022	13	14
94	NA	young	2023	13	21
95	NA	young	2022	14	5
96	NA	young	2023	14	31
97	NA	young	2022	15	35
98	NA	young	2023	15	23
99	NA	young	2022	16	0
100	NA	young	2023	16	33
101	NA	young	2022	17	13
102	NA	young	2023	17	19
103	NA	young	2022	1	0
104	NA	young	2023	1	22
105	NA	young	2022	2	14
106	NA	young	2023	2	21
107	NA	young	2022	3	17
108	NA	young	2023	3	11
109	NA	young	2022	4	2
110	NA	young	2023	4	16
111	NA	young	2022	5	4
112	NA	young	2023	5	5
113	NA	young	2022	6	35
114	NA	young	2023	6	6
115	NA	young	2022	7	17
116	NA	young	2023	7	18
117	NA	young	2022	8	3
118	NA	young	2023	8	1
119	NA	young	2022	9	3
120	NA	young	2023	9	31
121	NA	young	2022	10	20
122	NA	young	2023	10	17
123	NA	young	2022	11	9
124	NA	young	2023	11	31
125	NA	young	2022	12	29
126	NA	young	2023	12	10
127	NA	young	2022	13	15
128	NA	young	2023	13	7
129	NA	young	2022	14	26
130	NA	young	2023	14	35
131	NA	young	2022	15	26
132	NA	young	2023	15	14
133	NA	young	2022	16	17
134	NA	young	2023	16	18
135	NA	young	2022	17	30

136	NA	young	2023	17	12
137	NA	young	2022	1	17
138	NA	young	2023	1	20
139	NA	young	2022	2	2
140	NA	young	2023	2	13
141	NA	young	2022	3	33
142	NA	young	2023	3	20
143	NA	young	2022	4	28
144	NA	young	2023	4	35
145	NA	young	2022	5	3
146	NA	young	2023	5	11
147	NA	young	2022	6	2
148	NA	young	2023	6	7
149	NA	young	2022	7	25
150	NA	young	2023	7	32
151	NA	young	2022	8	2
152	NA	young	2023	8	18
153	NA	young	2022	9	18
154	NA	young	2023	9	4
155	NA	young	2022	10	32
156	NA	young	2023	10	18
157	NA	young	2022	11	7
158	NA	young	2023	11	13
159	NA	young	2022	12	2
160	NA	young	2023	12	26
161	NA	young	2022	13	6
162	NA	young	2023	13	17
163	NA	young	2022	14	28
164	NA	young	2023	14	0
165	NA	young	2022	15	11
166	NA	young	2023	15	20
167	NA	young	2022	16	9
168	NA	young	2023	16	6
169	NA	young	2022	17	1
170	NA	young	2023	17	35
171	NA	young	2022	1	29
172	NA	young	2023	1	13
173	NA	young	2022	2	16
174	NA	young	2023	2	32
175	NA	young	2022	3	19
176	NA	young	2023	3	14
177	NA	young	2022	4	27
178	NA	young	2023	4	6
179	NA	young	2022	5	17
180	NA	young	2023	5	18

181	NA	young	2022	6	25
182	NA	young	2023	6	19
183	NA	young	2022	7	35
184	NA	young	2023	7	34
185	NA	young	2022	8	14
186	NA	young	2023	8	3
187	NA	young	2022	9	27
188	NA	young	2023	9	19
189	NA	young	2022	10	14
190	NA	young	2023	10	23
191	NA	young	2022	11	24
192	NA	young	2023	11	16
193	NA	young	2022	12	6
194	NA	young	2023	12	29
195	NA	young	2022	13	12
196	NA	young	2023	13	22
197	NA	young	2022	14	32
198	NA	young	2023	14	4
199	NA	young	2022	15	1
200	NA	young	2023	15	34
201	NA	young	2022	16	23
202	NA	young	2023	16	7
203	NA	young	2022	17	15
204	NA	young	2023	17	7
205	NA	young	2022	1	10
206	NA	young	2023	1	16
207	NA	young	2022	2	9
208	NA	young	2023	2	33
209	NA	young	2022	3	15
210	NA	young	2023	3	27
211	NA	young	2022	4	6
212	NA	young	2023	4	20
213	NA	young	2022	5	2
214	NA	young	2023	5	22
215	NA	young	2022	6	29
216	NA	young	2023	6	28
217	NA	young	2022	7	4
218	NA	young	2023	7	35
219	NA	young	2022	8	15
220	NA	young	2023	8	23
221	NA	young	2022	9	24
222	NA	young	2023	9	32
223	NA	young	2022	10	16
224	NA	young	2023	10	25
225	NA	young	2022	11	28

226	NA	young	2023	11	21
227	NA	young	2022	12	8
228	NA	young	2023	12	14
229	NA	young	2022	13	16
230	NA	young	2023	13	29
231	NA	young	2022	14	34
232	NA	young	2023	14	28
233	NA	young	2022	15	24
234	NA	young	2023	15	19
235	NA	young	2022	16	27
236	NA	young	2023	16	29
237	NA	young	2022	17	31
238	NA	young	2023	17	35
239	NA	old	2022	1	3
240	NA	old	2023	1	17
241	NA	old	2022	2	34
242	NA	old	2023	2	12
243	NA	old	2022	3	4
244	NA	old	2023	3	26
245	NA	old	2022	4	17
246	NA	old	2023	4	17
247	NA	old	2022	5	12
248	NA	old	2023	5	18
249	NA	old	2022	6	10
250	NA	old	2023	6	13
251	NA	old	2022	7	18
252	NA	old	2023	7	16
253	NA	old	2022	8	12
254	NA	old	2023	8	10
255	NA	old	2022	9	7
256	NA	old	2023	9	25
257	NA	old	2022	10	9
258	NA	old	2023	10	8
259	NA	old	2022	11	5
260	NA	old	2023	11	19
261	NA	old	2022	12	12
262	NA	old	2023	12	20
263	NA	old	2022	13	26
264	NA	old	2023	13	12
265	NA	old	2022	14	5
266	NA	old	2023	14	33
267	NA	old	2022	15	20
268	NA	old	2023	15	20
269	NA	old	2022	16	11
270	NA	old	2023	16	2

271	NA	old	2022	17	34
272	NA	old	2023	17	13
273	NA	young	2022	1	28
274	NA	young	2023	1	22
275	NA	young	2022	2	21
276	NA	young	2023	2	4
277	NA	young	2022	3	7
278	NA	young	2023	3	8
279	NA	young	2022	4	34
280	NA	young	2023	4	10
281	NA	young	2022	5	0
282	NA	young	2023	5	15
283	NA	young	2022	6	9
284	NA	young	2023	6	3
285	NA	young	2022	7	14
286	NA	young	2023	7	34
287	NA	young	2022	8	17
288	NA	young	2023	8	20
289	NA	young	2022	9	31
290	NA	young	2023	9	0
291	NA	young	2022	10	32
292	NA	young	2023	10	22
293	NA	young	2022	11	15
294	NA	young	2023	11	33
295	NA	young	2022	12	6
296	NA	young	2023	12	10
297	NA	young	2022	13	14
298	NA	young	2023	13	21
299	NA	young	2022	14	18
300	NA	young	2023	14	30
301	NA	young	2022	15	20
302	NA	young	2023	15	22
303	NA	young	2022	16	34
304	NA	young	2023	16	2
305	NA	young	2022	17	24
306	NA	young	2023	17	7
307	NA	young	2022	1	34
308	NA	young	2023	1	18
309	NA	young	2022	2	21
310	NA	young	2023	2	8
311	NA	young	2022	3	16
312	NA	young	2023	3	26
313	NA	young	2022	4	17
314	NA	young	2023	4	5
315	NA	young	2022	5	18

316	NA	young	2023	5	4
317	NA	young	2022	6	34
318	NA	young	2023	6	13
319	NA	young	2022	7	13
320	NA	young	2023	7	6
321	NA	young	2022	8	20
322	NA	young	2023	8	22
323	NA	young	2022	9	4
324	NA	young	2023	9	13
325	NA	young	2022	10	24
326	NA	young	2023	10	17
327	NA	young	2022	11	34
328	NA	young	2023	11	2
329	NA	young	2022	12	32
330	NA	young	2023	12	29
331	NA	young	2022	13	31
332	NA	young	2023	13	16
333	NA	young	2022	14	10
334	NA	young	2023	14	15
335	NA	young	2022	15	16
336	NA	young	2023	15	15
337	NA	young	2022	16	34
338	NA	young	2023	16	5
339	NA	young	2022	17	29
340	NA	young	2023	17	8
341	NA	<NA>	2022	1	24
342	NA	<NA>	2023	1	13
343	NA	<NA>	2022	2	11
344	NA	<NA>	2023	2	5
345	NA	<NA>	2022	3	27
346	NA	<NA>	2023	3	32
347	NA	<NA>	2022	4	16
348	NA	<NA>	2023	4	27
349	NA	<NA>	2022	5	2
350	NA	<NA>	2023	5	23
351	NA	<NA>	2022	6	11
352	NA	<NA>	2023	6	12
353	NA	<NA>	2022	7	26
354	NA	<NA>	2023	7	29
355	NA	<NA>	2022	8	18
356	NA	<NA>	2023	8	15
357	NA	<NA>	2022	9	24
358	NA	<NA>	2023	9	25
359	NA	<NA>	2022	10	35
360	NA	<NA>	2023	10	20

361	NA	<NA>	2022	11	29
362	NA	<NA>	2023	11	0
363	NA	<NA>	2022	12	20
364	NA	<NA>	2023	12	23
365	NA	<NA>	2022	13	5
366	NA	<NA>	2023	13	27
367	NA	<NA>	2022	14	9
368	NA	<NA>	2023	14	27
369	NA	<NA>	2022	15	29
370	NA	<NA>	2023	15	17
371	NA	<NA>	2022	16	5
372	NA	<NA>	2023	16	2
373	NA	<NA>	2022	17	22
374	NA	<NA>	2023	17	1
375	NA	old	2022	1	21
376	NA	old	2023	1	7
377	NA	old	2022	2	24
378	NA	old	2023	2	32
379	NA	old	2022	3	17
380	NA	old	2023	3	6
381	NA	old	2022	4	30
382	NA	old	2023	4	4
383	NA	old	2022	5	12
384	NA	old	2023	5	17
385	NA	old	2022	6	35
386	NA	old	2023	6	15
387	NA	old	2022	7	28
388	NA	old	2023	7	11
389	NA	old	2022	8	22
390	NA	old	2023	8	1
391	NA	old	2022	9	11
392	NA	old	2023	9	12
393	NA	old	2022	10	23
394	NA	old	2023	10	9
395	NA	old	2022	11	29
396	NA	old	2023	11	25
397	NA	old	2022	12	30
398	NA	old	2023	12	4
399	NA	old	2022	13	30
400	NA	old	2023	13	17
401	NA	old	2022	14	32
402	NA	old	2023	14	33
403	NA	old	2022	15	2
404	NA	old	2023	15	19
405	NA	old	2022	16	11

```
406         NA          old   2023   16             2
407         NA          old   2022   17            16
408         NA          old   2023   17            30
```

```r
dim(dataLong)
```

```
[1] 408  12
```

```r
names(dataLong)
```

```
 [1] "ID"            "name"          "position"      "newVar1"
 [5] "newVar2"       "factorVar"     "numericVar"    "integerVar"
 [9] "characterVar"  "season"        "week"          "fantasyPoints"
```

Below, we widen the data by two variables (`season` and `week`), using `tidyverse`, so that the data are now in "`player` form" (where each row is uniquely identified by the `ID` variable):

```r
dataWide <- dataLong %>%
  pivot_wider(
    names_from = c(season, week),
    names_glue = "{.value}_{season}_week{week}",
    values_from = fantasyPoints)

dataWide
```

```
# A tibble: 12 x 43
    ID name        position newVar1 newVar2 factorVar numericVar integerVar
  <int> <chr>        <chr>   <chr>   <chr>   <fct>        <dbl>     <int>
 1     1 Ken Cussion    QB      quarte~ old     1              40        NA
 2     2 Ben Sacked     QB      quarte~ old     2              30        NA
 3     3 Chuck Downfie~ QB      quarte~ young   3              24        NA
 4     4 Ron Ingback    RB      runnin~ young   4              20        NA
 5     5 Rhonda Ball    RB      runnin~ young   5              18        NA
 6     6 Hugo Long      WR      wide r~ young   6              23        NA
 7     7 Lionel Scrimm~ WR      wide r~ young   7              27        NA
 8     8 Drew Blood     WR      wide r~ old     8              32        NA
 9     9 Chase Emdown   WR      wide r~ young   9              26        NA
10    10 Justin Time    TE      tight ~ young   10             23        NA
11    11 Spike D'Ball   TE      tight ~ <NA>    11             NA        NA
12    12 Isac Ulooz     LB      <NA>    old     12             37        NA
# i 35 more variables: characterVar <chr>, fantasyPoints_2022_week1 <int>,
#   fantasyPoints_2023_week1 <int>, fantasyPoints_2022_week2 <int>,
```

```
#   fantasyPoints_2023_week2 <int>, fantasyPoints_2022_week3 <int>,
#   fantasyPoints_2023_week3 <int>, fantasyPoints_2022_week4 <int>,
#   fantasyPoints_2023_week4 <int>, fantasyPoints_2022_week5 <int>,
#   fantasyPoints_2023_week5 <int>, fantasyPoints_2022_week6 <int>,
#   fantasyPoints_2023_week6 <int>, fantasyPoints_2022_week7 <int>, ...
```

```r
dim(dataWide)
```

```
[1] 12 43
```

```r
names(dataWide)
```

```
 [1] "ID"                        "name"
 [3] "position"                  "newVar1"
 [5] "newVar2"                   "factorVar"
 [7] "numericVar"                "integerVar"
 [9] "characterVar"              "fantasyPoints_2022_week1"
[11] "fantasyPoints_2023_week1"  "fantasyPoints_2022_week2"
[13] "fantasyPoints_2023_week2"  "fantasyPoints_2022_week3"
[15] "fantasyPoints_2023_week3"  "fantasyPoints_2022_week4"
[17] "fantasyPoints_2023_week4"  "fantasyPoints_2022_week5"
[19] "fantasyPoints_2023_week5"  "fantasyPoints_2022_week6"
[21] "fantasyPoints_2023_week6"  "fantasyPoints_2022_week7"
[23] "fantasyPoints_2023_week7"  "fantasyPoints_2022_week8"
[25] "fantasyPoints_2023_week8"  "fantasyPoints_2022_week9"
[27] "fantasyPoints_2023_week9"  "fantasyPoints_2022_week10"
[29] "fantasyPoints_2023_week10" "fantasyPoints_2022_week11"
[31] "fantasyPoints_2023_week11" "fantasyPoints_2022_week12"
[33] "fantasyPoints_2023_week12" "fantasyPoints_2022_week13"
[35] "fantasyPoints_2023_week13" "fantasyPoints_2022_week14"
[37] "fantasyPoints_2023_week14" "fantasyPoints_2022_week15"
[39] "fantasyPoints_2023_week15" "fantasyPoints_2022_week16"
[41] "fantasyPoints_2023_week16" "fantasyPoints_2022_week17"
[43] "fantasyPoints_2023_week17"
```

3.23 Transform Data from Wide to Long

Conversely, we can also restructure data from wide to long. Here are the data in long form, after they have been transformed from wide form using `tidyverse`:

```r
dataLong <- dataWide %>%
  pivot_longer(
    cols = fantasyPoints_2022_week1:fantasyPoints_2023_week17,
    names_to = c("season", "week"),
    names_pattern = "fantasyPoints_(.*)_week(.*)",
    values_to = "fantasyPoints")

dataLong
```

```
# A tibble: 408 x 12
    ID name      position newVar1   newVar2 factorVar numericVar integerVar
  <int> <chr>     <chr>   <chr>     <chr>   <fct>        <dbl>     <int>
 1    1 Ken Cussion QB      quarterba~ old    1              40        NA
 2    1 Ken Cussion QB      quarterba~ old    1              40        NA
 3    1 Ken Cussion QB      quarterba~ old    1              40        NA
 4    1 Ken Cussion QB      quarterba~ old    1              40        NA
 5    1 Ken Cussion QB      quarterba~ old    1              40        NA
 6    1 Ken Cussion QB      quarterba~ old    1              40        NA
 7    1 Ken Cussion QB      quarterba~ old    1              40        NA
 8    1 Ken Cussion QB      quarterba~ old    1              40        NA
 9    1 Ken Cussion QB      quarterba~ old    1              40        NA
10    1 Ken Cussion QB      quarterba~ old    1              40        NA
# i 398 more rows
# i 4 more variables: characterVar <chr>, season <chr>, week <chr>,
#   fantasyPoints <int>
```

```r
dim(dataLong)
```

```
[1] 408  12
```

```r
names(dataLong)
```

```
 [1] "ID"            "name"          "position"      "newVar1"
 [5] "newVar2"       "factorVar"     "numericVar"    "integerVar"
 [9] "characterVar"  "season"        "week"          "fantasyPoints"
```

3.24 Loops

If you want to perform the same computation multiple times, it can be faster to do it in a loop compared to writing out the same computation many times.

For instance, here is a loop that runs from 1 to 12 (the number of players in the `players` object), incrementing by 1 after each iteration. The loop prints each element of a vector (i.e., the player's name) and the loop index (`i`) that indicates where the loop is in terms of its iterations:

```
for(i in 1:length(players$ID)){
  print(paste("The loop is at index:", i, sep = " "))
  print(paste("My favorite player is:", players$name[i], sep = " "))
}
```

```
[1] "The loop is at index: 1"
[1] "My favorite player is: Ken Cussion"
[1] "The loop is at index: 2"
[1] "My favorite player is: Ben Sacked"
[1] "The loop is at index: 3"
[1] "My favorite player is: Chuck Downfield"
[1] "The loop is at index: 4"
[1] "My favorite player is: Ron Ingback"
[1] "The loop is at index: 5"
[1] "My favorite player is: Rhonda Ball"
[1] "The loop is at index: 6"
[1] "My favorite player is: Hugo Long"
[1] "The loop is at index: 7"
[1] "My favorite player is: Lionel Scrimmage"
[1] "The loop is at index: 8"
[1] "My favorite player is: Drew Blood"
[1] "The loop is at index: 9"
[1] "My favorite player is: Chase Emdown"
[1] "The loop is at index: 10"
[1] "My favorite player is: Justin Time"
[1] "The loop is at index: 11"
[1] "My favorite player is: Spike D'Ball"
[1] "The loop is at index: 12"
[1] "My favorite player is: Isac Ulooz"
```

4

Download and Process NFL Football Data

4.1 Load Packages

```
library("ffanalytics")
library("petersenlab")
library("nflreadr")
library("nflfastR")
library("nfl4th")
library("nflplotR")
library("progressr")
library("lubridate")
library("tidyverse")
```

4.2 Data Dictionaries of NFL Data

Data Dictionaries are metadata that describe the meaning of the variables in a dataset. You can find Data Dictionaries for the various National Football League (NFL) datasets at the following link: https://nflreadr.nflverse.com/articles/index.html.

4.3 Types of NFL Data

Below, we provide examples for how to download various types of NFL data. For additional resources, Congelio (2023) provides a helpful introductory text for working with NFL data in `R`. We save each data file after downloading it, so we can use the data in subsequent chapters. This chapter uses extensive use of merging to process the data for later use. See Section 3.21 for a reminder of how to perform merging, the types of merges, and what you can expect when you merge data objects with different formats. Guidance for how to merge the various NFL-related data files is provided at the following link: https://github.com/nflverse/nfldata/blob/master/DATASETS.md.

4.3.1 Players

```
nfl_players_raw <- progressr::with_progress(
  nflreadr::load_players())
```

```
save(
  nfl_players_raw,
  file = "./data/nfl_players_raw.RData"
)
```

The `nfl_players` object is in `player` form. That is, each row should be uniquely identified by `gsis_id`. Let's rearrange the data accordingly:

```
nfl_players <- nfl_players_raw %>%
  select(gsis_id, everything()) %>%
  arrange(display_name)
```

Let's check for duplicate `player` instances:

```
nfl_players %>%
  group_by(gsis_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 33
# Groups:   gsis_id [0]
# i 33 variables: gsis_id <chr>, status <chr>, display_name <chr>,
```

```
#   first_name <chr>, last_name <chr>, esb_id <chr>, birth_date <chr>,
#   college_name <chr>, position_group <chr>, position <chr>,
#  jersey_number <int>, height <dbl>, weight <int>, years_of_experience <chr>,
#   team_abbr <chr>, team_seq <int>, current_team_id <chr>,
#  football_name <chr>, entry_year <int>, rookie_year <int>, draft_club <chr>,
#   draft_number <int>, college_conference <chr>, ...
```

4.3.1.1 Processing

Let's do some data cleanup:

```
# Convert missing values to NA
nfl_players[nfl_players == ""] <- NA

# Drop players with missing values for gsis_id
nfl_players <- nfl_players %>%
  filter(!is.na(gsis_id))
```

```
save(
  nfl_players,
  file = "./data/nfl_players.RData"
)
```

4.3.2 Teams

```
nfl_teams_raw <- progressr::with_progress(
  nflreadr::load_teams(current = TRUE))
```

```
save(
  nfl_teams_raw,
  file = "./data/nfl_teams_raw.RData"
)
```

The `nfl_teams` object is in `team` form. That is, each row should be uniquely identified by `team_id`. Let's rearrange the data accordingly:

```
nfl_teams <- nfl_teams_raw %>%
  select(team_id, everything())
```

Let's check for duplicate `team` instances:

```
nfl_teams %>%
  group_by(team_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 16
# Groups:   team_id [0]
# i 16 variables: team_id <chr>, team_abbr <chr>, team_name <chr>,
#  team_nick <chr>, team_conf <chr>, team_division <chr>, team_color <chr>,
#   team_color2 <chr>, team_color3 <chr>, team_color4 <chr>,
#  team_logo_wikipedia <chr>, team_logo_espn <chr>, team_wordmark <chr>,
#  team_conference_logo <chr>, team_league_logo <chr>, team_logo_squared <chr>
```

```
save(
  nfl_teams,
  file = "./data/nfl_teams.RData"
)
```

4.3.3 Fantasy Player IDs

A Data Dictionary for fantasy player ID data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_ff_playerids.html

```
nfl_playerIDs_raw <- progressr::with_progress(
  nflreadr::load_ff_playerids())
```

```
save(
  nfl_playerIDs_raw,
  file = "./data/nfl_playerIDs_raw.RData"
)
```

The `nfl_playerIDs` object is in `player` form. That is, each row should be uniquely identified by `mfl_id`.

```
nfl_playerIDs <- nfl_playerIDs_raw %>%
  arrange(name, mfl_id)
```

Let's check for duplicate `player` instances:

```
nfl_playerIDs %>%
  group_by(mfl_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 35
# Groups:   mfl_id [0]
# i 35 variables: mfl_id <chr>, sportradar_id <chr>, fantasypros_id <chr>,
#   gsis_id <chr>, pff_id <chr>, sleeper_id <chr>, nfl_id <chr>, espn_id <chr>,
#    yahoo_id <chr>, fleaflicker_id <chr>, cbs_id <chr>, pfr_id <chr>,
#   cfbref_id <chr>, rotowire_id <chr>, rotoworld_id <chr>, ktc_id <chr>,
#    stats_id <chr>, stats_global_id <chr>, fantasy_data_id <chr>,
#   swish_id <chr>, name <chr>, merge_name <chr>, position <chr>, team <chr>,
#   birthdate <date>, age <dbl>, draft_year <int>, draft_round <int>, ...
```

```
nfl_playerIDs %>%
  filter(!is.na(gsis_id)) %>%
  group_by(gsis_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 6 x 35
# Groups:   gsis_id [3]
  mfl_id sportradar_id  fantasypros_id gsis_id pff_id sleeper_id nfl_id espn_id
  <chr>  <chr>          <chr>          <chr>   <chr>  <chr>     <chr>  <chr>
1 12982  0f1b8946-54b9-~ 14554         00-003~ 10117  2697      <NA>   2578554
2 5738   0f1b8946-54b9-~ 14554         00-003~ 10117  2697      <NA>   2578554
3 1286   <NA>           <NA>           00-002~ 869    <NA>      <NA>   <NA>
4 5027   <NA>           <NA>           00-002~ 869    <NA>      <NA>   2670
5 3298   <NA>           <NA>           00-001~ 333    <NA>      <NA>   1430
6 8058   <NA>           <NA>           00-001~ 333    <NA>      <NA>   <NA>
# i 27 more variables: yahoo_id <chr>, fleaflicker_id <chr>, cbs_id <chr>,
#   pfr_id <chr>, cfbref_id <chr>, rotowire_id <chr>, rotoworld_id <chr>,
#   ktc_id <chr>, stats_id <chr>, stats_global_id <chr>, fantasy_data_id <chr>,
#   swish_id <chr>, name <chr>, merge_name <chr>, position <chr>, team <chr>,
#    birthdate <date>, age <dbl>, draft_year <int>, draft_round <int>,
#   draft_pick <int>, draft_ovr <chr>, twitter_username <chr>, height <int>,
#    weight <int>, college <chr>, db_season <int>
```

Let's do some data processing to help with merging the dataset with other datasets:

```
nfl_playerIDs$name[which(nfl_playerIDs$name == "Bennett,Michael")] <- "Michael Bennett"

nfl_playerIDs <- nfl_playerIDs %>%
  mutate(
    merge_name = nflreadr::clean_player_names(name, lowercase = TRUE)
  )
```

```
save(
  nfl_playerIDs,
  file = "./data/nfl_playerIDs.RData"
)
```

4.3.4 Player Info

4.3.5 Rosters

A Data Dictionary for rosters is located at the following links:

- https://nflreadr.nflverse.com/articles/dictionary_rosters.html
- https://github.com/nflverse/nfldata/blob/master/DATASETS.md#rosters

```
nfl_rosters_raw <- progressr::with_progress(
  nflreadr::load_rosters(seasons = TRUE))

nfl_rosters_weekly_raw <- progressr::with_progress(
  nflreadr::load_rosters_weekly(seasons = TRUE))

rosters <- read_csv("https://raw.githubusercontent.com/leesharpe/nfldata/master/data/rosters.csv")
```

```
save(
  nfl_rosters_raw,
  rosters,
  file = "./data/nfl_rosters_raw.RData"
)

save(
  nfl_rosters_weekly_raw,
  file = "./data/nfl_rosters_weekly_raw.RData"
)
```

The `nfl_rosters` object is in `player-season-team` form. That is, each row should be uniquely identified by the combination of `gsis_id`, `season`, and `team`. Let's rearrange the data accordingly:

```
nfl_rosters <- nfl_rosters_raw %>%
  left_join(
    rosters %>% select(playerid, season, team, side, category, games, starts, years, av),
    by = c("season","team","pfr_id" = "playerid"),
```

```
    na_matches = "never"
  ) %>%
  select(gsis_id, season, team, week, everything()) %>%
  arrange(full_name, gsis_id, season, team, week)
```

Let's check for duplicate `player-season-team` instances:

```
nfl_rosters %>%
  group_by(gsis_id, season, team) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 6 x 42
# Groups:   gsis_id, season, team [4]
  gsis_id season team  week position depth_chart_position jersey_number status
  <chr>   <dbl> <chr> <int> <chr>    <chr>              <chr>        <chr>
1 <NA>     1922 HAM      NA RB       WB                  0            ACT
2 <NA>     1921 TON      NA OL       C                   0            ACT
3 <NA>     1920 HAM      NA OL       G                   0            ACT
4 <NA>     1921 TON      NA RB       BB                  0            ACT
5 <NA>     1920 DET      NA OL       G                   0            ACT
6 <NA>     1920 DET      NA RB       TB                  0            ACT
# i 34 more variables: full_name <chr>, first_name <chr>, last_name <chr>,
#   birth_date <date>, height <dbl>, weight <int>, college <chr>,
#  espn_id <chr>, sportradar_id <chr>, yahoo_id <chr>, rotowire_id <chr>,
#   pff_id <chr>, pfr_id <chr>, fantasy_data_id <chr>, sleeper_id <chr>,
#   years_exp <int>, headshot_url <chr>, esb_id <chr>, gsis_it_id <chr>,
#  smart_id <chr>, entry_year <int>, rookie_year <int>, draft_club <chr>,
#  ngs_position <chr>, game_type <chr>, status_description_abbr <chr>, ...
```

4.3.5.1 Processing

Let's do some data cleanup:

```
# Drop players with missing values for gsis_id
nfl_rosters <- nfl_rosters %>%
  filter(!is.na(gsis_id))

# Fill in missing values for a player in their duplicate instances, and then keep only the first o
nfl_rosters <- nfl_rosters %>%
  group_by(gsis_id, season, team) %>%
  fill(names(.), .direction = "downup") %>%
  slice_head(n = 1) %>%
  ungroup()
```

Let's check again for duplicate `player-season-team` instances:

```
nfl_rosters %>%
  group_by(gsis_id, season, team) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 42
# Groups:   gsis_id, season, team [0]
# i 42 variables: gsis_id <chr>, season <dbl>, team <chr>, week <int>,
#   position <chr>, depth_chart_position <chr>, jersey_number <chr>,
#   status <chr>, full_name <chr>, first_name <chr>, last_name <chr>,
#   birth_date <date>, height <dbl>, weight <int>, college <chr>,
#  espn_id <chr>, sportradar_id <chr>, yahoo_id <chr>, rotowire_id <chr>,
#   pff_id <chr>, pfr_id <chr>, fantasy_data_id <chr>, sleeper_id <chr>,
#  years_exp <int>, headshot_url <chr>, esb_id <chr>, gsis_it_id <chr>, ...
```

The `nfl_rosters_weekly` object is in `player-season-week` form. That is, each row should be uniquely identified by the combination of `gsis_id`, `season`, and `week`. Let's rearrange the data accordingly:

```
nfl_rosters_weekly <- nfl_rosters_weekly_raw %>%
  select(gsis_id, season, week, everything()) %>%
  arrange(full_name, gsis_id, season, week)
```

Let's check for duplicate `player-season-week` instances:

```
nfl_rosters_weekly %>%
  group_by(gsis_id, season, week) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 6 x 36
# Groups:   gsis_id, season, week [3]
  gsis_id  season  week team  position depth_chart_position jersey_number status
  <chr>    <int> <int> <chr> <chr>    <chr>             <chr>        <chr>
1 00-0027~   2011     1 NE    LB       <NA>               90           TRD
2 00-0027~   2011     1 NE    LB       <NA>               90           ACT
3 00-0027~   2011     2 NE    LB       <NA>               90           TRD
4 00-0027~   2011     2 NE    LB       <NA>               90           ACT
5 00-0027~   2011     4 IND   LB       <NA>               52           ACT
6 00-0027~   2011     4 IND   LB       <NA>               52           TRD
# i 28 more variables: full_name <chr>, first_name <chr>, last_name <chr>,
#   birth_date <date>, height <dbl>, weight <int>, college <chr>,
```

```
#  espn_id <chr>, sportradar_id <chr>, yahoo_id <chr>, rotowire_id <chr>,
#   pff_id <chr>, pfr_id <chr>, fantasy_data_id <chr>, sleeper_id <chr>,
#  years_exp <int>, headshot_url <chr>, ngs_position <chr>, game_type <chr>,
#   status_description_abbr <chr>, football_name <chr>, esb_id <chr>,
#  gsis_it_id <chr>, smart_id <chr>, entry_year <int>, rookie_year <int>, ...
```

Let's do some data cleanup:

```
# Drop players with missing values for gsis_id
nfl_rosters_weekly <- nfl_rosters_weekly %>%
  filter(!is.na(gsis_id))

# Fill in missing values for a player in their duplicate instances, and then keep only the first o
nfl_rosters_weekly <- nfl_rosters_weekly %>%
  group_by(gsis_id, season, week) %>%
  fill(names(.), .direction = "downup") %>%
  slice_head(n = 1) %>%
  ungroup()
```

Let's check again for duplicate `player-season-week` instances:

```
nfl_rosters_weekly %>%
  group_by(gsis_id, season, week) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 36
# Groups:   gsis_id, season, week [0]
# i 36 variables: gsis_id <chr>, season <int>, week <int>, team <chr>,
#   position <chr>, depth_chart_position <chr>, jersey_number <chr>,
#   status <chr>, full_name <chr>, first_name <chr>, last_name <chr>,
#   birth_date <date>, height <dbl>, weight <int>, college <chr>,
#  espn_id <chr>, sportradar_id <chr>, yahoo_id <chr>, rotowire_id <chr>,
#   pff_id <chr>, pfr_id <chr>, fantasy_data_id <chr>, sleeper_id <chr>,
#   years_exp <int>, headshot_url <chr>, ngs_position <chr>, ...
```

```
save(
  nfl_rosters,
  file = "./data/nfl_rosters.RData"
)

save(
  nfl_rosters_weekly,
  file = "./data/nfl_rosters_weekly.RData"
)
```

4.3.6 Game Schedules

A Data Dictionary for game schedules data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_schedules.html

```
nfl_schedules_raw <- progressr::with_progress(
  nflreadr::load_schedules(seasons = TRUE))
```

```
save(
  nfl_schedules_raw,
  file = "./data/nfl_schedules_raw.RData"
)
```

The `nfl_schedules` object is in `game` form and in `season-week` (and `-game type`) form. That is, each row should be uniquely identified by `game_id`. Each row should also be uniquely identified by the combination of `season` and `week` (and `game type`).

```
nfl_schedules <- nfl_schedules_raw
```

Let's check for duplicate `game` instances:

```
nfl_schedules %>%
  group_by(game_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 46
# Groups:   game_id [0]
# i 46 variables: game_id <chr>, season <int>, game_type <chr>, week <int>,
#   gameday <chr>, weekday <chr>, gametime <chr>, away_team <chr>,
#   away_score <int>, home_team <chr>, home_score <int>, location <chr>,
#   result <int>, total <int>, overtime <int>, old_game_id <chr>, gsis <int>,
#   nfl_detail_id <chr>, pfr <chr>, pff <int>, espn <chr>, ftn <int>,
#   away_rest <int>, home_rest <int>, away_moneyline <int>,
#   home_moneyline <int>, spread_line <dbl>, away_spread_odds <int>, ...
```

```
save(
  nfl_schedules,
  file = "./data/nfl_schedules.RData"
)
```

4.3.7 Standings

A Data Dictionary for game schedules data is located at the following link: https://github.com/nflverse/nfldata/blob/master/DATASETS.md#standings

```
nfl_standings_raw <- read_csv("http://www.habitatring.com/standings.csv")
```

```
save(
  nfl_standings_raw,
  file = "./data/nfl_standings_raw.RData"
)
```

The `nfl_standings` object is in `season-team` form. That is, each row should be uniquely identified by the combination of `season` and `team`.

```
nfl_standings <- nfl_standings_raw
```

Let's check for duplicate `season-team` instances:

```
nfl_standings %>%
  group_by(season, team) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 16
# Groups:   season, team [0]
# i 16 variables: season <dbl>, conf <chr>, division <chr>, team <chr>,
#   wins <dbl>, losses <dbl>, ties <dbl>, pct <dbl>, div_rank <dbl>,
#  scored <dbl>, allowed <dbl>, net <dbl>, sov <dbl>, sos <dbl>, seed <dbl>,
#   playoff <chr>
```

```
save(
  nfl_standings,
  file = "./data/nfl_standings.RData"
)
```

4.3.8 The Combine

A Data Dictionary for data from the combine is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_combine.html

```
nfl_combine_raw <- progressr::with_progress(
  nflreadr::load_combine(seasons = TRUE))
```

```
save(
  nfl_combine_raw,
  file = "./data/nfl_combine_raw.RData"
)
```

The `nfl_combine` object is in `player` form. That is, each row should be uniquely identified by the player's `id`. However, there is no `gsis_id` variable to merge it easily with other datasets. Some of the players have other `id` variables, including `pfr_id` and `cfb_id`. Let's rearrange the data accordingly:

```
nfl_combine <- nfl_combine_raw %>%
  select(pfr_id, cfb_id, everything()) %>%
  arrange(season, player_name)
```

Let's do some data processing to help with merging the dataset with other datasets:

```
nfl_combine <- nfl_combine %>%
  mutate(
    merge_name = nflreadr::clean_player_names(player_name, lowercase = TRUE)
  )

# First, merge on both pfr_id and cfb_id
merged_data1 <- left_join(
  nfl_combine,
  nfl_playerIDs %>% select(pfr_id, cfbref_id, gsis_id),
  by = c("pfr_id", "cfb_id" = "cfbref_id"),
  na_matches = "never"
)

# Second, merge on pfr_id
merged_data2 <- left_join(
  nfl_combine,
  nfl_playerIDs %>% select(pfr_id, merge_name, position, gsis_id),
  by = c("pfr_id","merge_name","pos" = "position"),
  na_matches = "never"
)

# Third, merge on cfb_id
merged_data3 <- left_join(
  nfl_combine,
```

```
  nfl_playerIDs %>% select(cfbref_id, merge_name, position, gsis_id),
  by = c("cfb_id" = "cfbref_id","merge_name","pos" = "position"),
  na_matches = "never"
)

# Combine gsis_id across merges
nfl_combine$gsis_id <- coalesce(
  merged_data1$gsis_id,
  merged_data2$gsis_id,
  merged_data3$gsis_id
  )

# Rearrange the data
nfl_combine <- nfl_combine %>%
  select(gsis_id, pfr_id, cfb_id, player_name, everything()) %>%
  arrange(season, player_name)
```

Let's check for duplicate `gsis_id`, `pfr_id`, and `cfb_id` instances:

```
nfl_combine %>%
  group_by(gsis_id) %>%
  filter(n() > 1, !is.na(gsis_id)) %>%
  arrange(gsis_id) %>%
  head()
```

```
# A tibble: 4 x 20
# Groups:   gsis_id [2]
 gsis_id   pfr_id  cfb_id player_name season draft_year draft_team draft_round
 <chr>     <chr>   <chr>  <chr>       <int>     <dbl> <chr>          <dbl>
1 00-0023449 JohnDe~ derri~ Derrick Jo~  2005     2005 Kansas Ci~        1
2 00-0023449 JohnDe~ derri~ Derrick Jo~  2005     2005 Kansas Ci~        1
3 00-0035970 SamuSt~ stanf~ Stanford S~  2004       NA <NA>            NA
4 00-0035970 SamuSt~ stanf~ Stanford S~  2020       NA <NA>            NA
# i 12 more variables: draft_ovr <dbl>, pos <chr>, school <chr>, ht <chr>,
#   wt <dbl>, forty <dbl>, bench <dbl>, vertical <dbl>, broad_jump <dbl>,
#   cone <dbl>, shuttle <dbl>, merge_name <chr>
```

```
nfl_combine %>%
  group_by(pfr_id) %>%
  filter(n() > 1, !is.na(pfr_id)) %>%
  arrange(pfr_id) %>%
  head()
```

```
# A tibble: 6 x 20
```

```
# Groups:   pfr_id [3]
  gsis_id pfr_id   cfb_id   player_name season draft_year draft_team draft_round
  <chr>   <chr>    <chr>    <chr>       <int>     <dbl> <chr>          <dbl>
1 <NA>    BrowCh03 chris-b~ Chris Brown  2001      2003 Tennessee~        3
2 <NA>    BrowCh03 chris-b~ Chris Brown  2003      2003 Tennessee~        3
3 <NA>    BrowPh00 corey-b~ Corey Brown  2001        NA <NA>             NA
4 <NA>    BrowPh00 corey-b~ Corey Brown  2014        NA <NA>             NA
5 <NA>    CartCh00 <NA>     Chris Cart~  2010      2011 Pittsburg~        5
6 <NA>    CartCh00 chris-c~ Chris Cart~  2011      2011 Pittsburg~        5
# i 12 more variables: draft_ovr <dbl>, pos <chr>, school <chr>, ht <chr>,
#   wt <dbl>, forty <dbl>, bench <dbl>, vertical <dbl>, broad_jump <dbl>,
#   cone <dbl>, shuttle <dbl>, merge_name <chr>
```

```r
nfl_combine %>%
  group_by(cfb_id) %>%
  filter(n() > 1, !is.na(cfb_id)) %>%
  arrange(cfb_id) %>%
  head()
```

```
# A tibble: 6 x 20
# Groups:   cfb_id [3]
  gsis_id   pfr_id  cfb_id player_name season draft_year draft_team draft_round
  <chr>     <chr>   <chr>  <chr>       <int>     <dbl> <chr>          <dbl>
1 <NA>      DaviBu~ buste~ Buster Dav~  2007      2007 Arizona C~        3
2 <NA>      DaviBu~ buste~ Buster Dav~  2007      2007 Arizona C~        3
3 00-0023449 JohnDe~ derri~ Derrick Jo~  2005      2005 Kansas Ci~        1
4 00-0023449 JohnDe~ derri~ Derrick Jo~  2005      2005 Kansas Ci~        1
5 <NA>      <NA>    jarre~ Jarrett Pa~  2021        NA <NA>             NA
6 <NA>      PattJa~ jarre~ Jarrett Pa~  2023      2023 Houston T~        6
# i 12 more variables: draft_ovr <dbl>, pos <chr>, school <chr>, ht <chr>,
#   wt <dbl>, forty <dbl>, bench <dbl>, vertical <dbl>, broad_jump <dbl>,
#   cone <dbl>, shuttle <dbl>, merge_name <chr>
```

Let's do some additional data processing:

```r
# Drop Stanford Samuels Jr.
nfl_combine$gsis_id[which(nfl_combine$cfb_id == "stanford-samuels-1")] <- NA

nfl_combine <- nfl_combine %>%
  separate_wider_delim(
    ht,
    names = c("feet", "inches"),
    delim = "-") %>%
  mutate(
```

```
    feet = as.numeric(feet),
    inches = as.numeric(inches),
    ht = feet * 12 + inches
  ) %>%
  select(-feet, -inches)
```

However, these apparent duplicates appear to be different players at different positions:

```
nfl_combine %>%
  group_by(season, gsis_id, pos) %>%
  filter(n() > 1, !is.na(gsis_id)) %>%
  arrange(gsis_id) %>%
  head()
```

```
# A tibble: 0 x 20
# Groups:   season, gsis_id, pos [0]
# i 20 variables: gsis_id <chr>, pfr_id <chr>, cfb_id <chr>, player_name <chr>,
#   season <int>, draft_year <dbl>, draft_team <chr>, draft_round <dbl>,
#   draft_ovr <dbl>, pos <chr>, school <chr>, wt <dbl>, forty <dbl>,
#  bench <dbl>, vertical <dbl>, broad_jump <dbl>, cone <dbl>, shuttle <dbl>,
#   merge_name <chr>, ht <dbl>
```

```
nfl_combine %>%
  group_by(season, pfr_id, pos) %>%
  filter(n() > 1, !is.na(pfr_id)) %>%
  arrange(pfr_id) %>%
  head()
```

```
# A tibble: 0 x 20
# Groups:   season, pfr_id, pos [0]
# i 20 variables: gsis_id <chr>, pfr_id <chr>, cfb_id <chr>, player_name <chr>,
#   season <int>, draft_year <dbl>, draft_team <chr>, draft_round <dbl>,
#   draft_ovr <dbl>, pos <chr>, school <chr>, wt <dbl>, forty <dbl>,
#  bench <dbl>, vertical <dbl>, broad_jump <dbl>, cone <dbl>, shuttle <dbl>,
#   merge_name <chr>, ht <dbl>
```

```
nfl_combine %>%
  group_by(season, cfb_id, pos) %>%
  filter(n() > 1, !is.na(cfb_id)) %>%
  arrange(cfb_id) %>%
  head()
```

```
# A tibble: 0 x 20
# Groups:   season, cfb_id, pos [0]
# i 20 variables: gsis_id <chr>, pfr_id <chr>, cfb_id <chr>, player_name <chr>,
#   season <int>, draft_year <dbl>, draft_team <chr>, draft_round <dbl>,
#   draft_ovr <dbl>, pos <chr>, school <chr>, wt <dbl>, forty <dbl>,
#  bench <dbl>, vertical <dbl>, broad_jump <dbl>, cone <dbl>, shuttle <dbl>,
#   merge_name <chr>, ht <dbl>
```

```
save(
  nfl_combine,
  file = "./data/nfl_combine.RData"
)
```

4.3.9 Draft Picks

A Data Dictionary for draft picks data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_draft_picks.html

```
nfl_draftPicks_raw <- progressr::with_progress(
  nflreadr::load_draft_picks(seasons = TRUE))
```

```
save(
  nfl_draftPicks_raw,
  file = "./data/nfl_draftPicks_raw.RData"
)
```

The `nfl_draftPicks` object is in `player` form. That is, each row should be uniquely identified by `gsis_id`. Let's rearrange the data accordingly:

```
nfl_draftPicks <- nfl_draftPicks_raw %>%
  select(gsis_id, everything()) %>%
  arrange(pfr_player_name)
```

Let's check for duplicate `player` instances:

```
nfl_draftPicks %>%
  group_by(gsis_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 6 x 36
# Groups:   gsis_id [2]
```

```
  gsis_id season round  pick team  pfr_player_id cfb_player_id   pfr_player_name
  <chr>    <int> <int> <int> <chr> <chr>          <chr>            <chr>
1 <NA>     1995    5   162 SDG   ElliOm20     omar-ellison-1  'Omar Ellison
2 ""       1989    8   209 NYJ   BrowAB00      anthony-brown-7 A.B. Brown
3 <NA>     1995    7   231 CLE   <NA>         ac-tellison-1   A.C. Tellison
4 <NA>     2015    3    67 JAX   CannA.00      aj-cann-1       A.J. Cann
5 <NA>     2007    4   105 DET   <NA>          aj-davis-1      A.J. Davis
6 <NA>     2010    4   119 MIA   EddsA.99      <NA>            A.J. Edds
# i 28 more variables: hof <lgl>, position <chr>, category <chr>, side <chr>,
#   college <chr>, age <int>, to <int>, allpro <int>, probowls <int>,
#  seasons_started <int>, w_av <int>, car_av <lgl>, dr_av <int>, games <int>,
#   pass_completions <int>, pass_attempts <int>, pass_yards <int>,
#   pass_tds <int>, pass_ints <int>, rush_atts <int>, rush_yards <int>,
#   rush_tds <int>, receptions <int>, rec_yards <int>, rec_tds <int>,
#   def_solo_tackles <int>, def_ints <int>, def_sacks <dbl>
```

```
save(
  nfl_draftPicks,
  file = "./data/nfl_draftPicks.RData"
)
```

4.3.10 Draft Values

A Data Dictionary for draft values data is located at the following link: https://github.com/nflverse/nfldata/blob/master/DATASETS.md#draft_values

```
nfl_draftValues_raw <- read_csv("https://raw.githubusercontent.com/leesharpe/nfldata/master/data/d
```

```
save(
  nfl_draftValues_raw,
  file = "./data/nfl_draftValues_raw.RData"
)
```

The `nfl_draftValues` object is in `pick` form. That is, each row should be uniquely identified by `pick`. Let's rearrange the data accordingly:

```
nfl_draftValues <- nfl_draftValues_raw %>%
  select(pick, everything()) %>%
  arrange(pick)
```

Let's check for duplicate `pick` instances:

```r
nfl_draftValues %>%
  group_by(pick) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 6
# Groups:   pick [0]
# i 6 variables: pick <dbl>, stuart <dbl>, johnson <dbl>, hill <dbl>,
#   otc <dbl>, pff <dbl>
```

```r
save(
  nfl_draftValues,
  file = "./data/nfl_draftValues.RData"
)
```

4.3.10.1 Processing

Let's do some data cleanup:

```r
# Convert missing values to NA
nfl_draftPicks[nfl_draftPicks == ""] <- NA

# Drop players with missing values for gsis_id
nfl_draftPicks <- nfl_draftPicks %>%
  filter(!is.na(gsis_id))
```

Let's check again for duplicate `player` instances:

```r
nfl_draftPicks %>%
  group_by(gsis_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 36
# Groups:   gsis_id [0]
# i 36 variables: gsis_id <chr>, season <int>, round <int>, pick <int>,
#   team <chr>, pfr_player_id <chr>, cfb_player_id <chr>,
#   pfr_player_name <chr>, hof <lgl>, position <chr>, category <chr>,
#   side <chr>, college <chr>, age <int>, to <int>, allpro <int>,
#   probowls <int>, seasons_started <int>, w_av <int>, car_av <lgl>,
#  dr_av <int>, games <int>, pass_completions <int>, pass_attempts <int>,
#  pass_yards <int>, pass_tds <int>, pass_ints <int>, rush_atts <int>, ...
```

```
save(
  nfl_draftPicks,
  file = "./data/nfl_draftPicks.RData"
)
```

4.3.11 Depth Charts

A Data Dictionary for data from weekly depth charts is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_depth_charts.html

```
nfl_depthCharts_raw <- progressr::with_progress(
  nflreadr::load_depth_charts(seasons = TRUE))
```

```
save(
  nfl_depthCharts_raw,
  file = "./data/nfl_depthCharts_raw.RData"
)
```

The `nfl_depthCharts` object is in `player-season-week-position` form. That is, each row should be uniquely identified by the combination of `gsis_id`, `season`, `week`, and `depth_position`. Let's rearrange the data accordingly:

```
nfl_depthCharts <- nfl_depthCharts_raw %>%
  select(gsis_id, season, week, depth_position, everything()) %>%
  arrange(full_name, gsis_id, season, week, depth_position)
```

Let's check for duplicate `player-season-week-position` instances:

```
nfl_depthCharts %>%
  group_by(gsis_id, season, week, depth_position) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 6 x 15
# Groups:   gsis_id, season, week, depth_position [3]
  gsis_id   season  week depth_position club_code game_type depth_team last_name
  <chr>      <dbl> <dbl> <chr>         <chr>     <chr>     <chr>      <chr>
1 00-00328~   2021     5 DE            LA        REG       1          Robinson
2 00-00328~   2021     5 DE            LA        POST      1          Robinson
3 00-00356~   2022     5 WR            PHI       REG       1          Brown
4 00-00356~   2022     5 WR            PHI       POST      1          Brown
5 00-00242~   2010     5 BLB           GB        REG       1          Hawk
```

```
6 00-00242~   2010     5 BLB           GB        POST      1          Hawk
# i 7 more variables: first_name <chr>, football_name <chr>, formation <chr>,
#   jersey_number <chr>, position <chr>, elias_id <chr>, full_name <chr>
```

4.3.11.1 Processing

Let's do some data cleanup:

```
# Drop players with missing values for gsis_id
nfl_depthCharts <- nfl_depthCharts %>%
  filter(!is.na(gsis_id))

# Fill in missing values for a player in their duplicate instances, and then keep only the first o
nfl_depthCharts <- nfl_depthCharts %>%
  group_by(gsis_id, season, week, depth_position) %>%
  fill(names(.), .direction = "downup") %>%
  slice_head(n = 1) %>%
  ungroup()
```

Let's check again for duplicate `player-season-week-position` instances:

```
nfl_depthCharts %>%
  group_by(gsis_id, season, week, depth_position) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 15
# Groups:   gsis_id, season, week, depth_position [0]
# i 15 variables: gsis_id <chr>, season <dbl>, week <dbl>,
#  depth_position <chr>, club_code <chr>, game_type <chr>, depth_team <chr>,
#  last_name <chr>, first_name <chr>, football_name <chr>, formation <chr>,
#   jersey_number <chr>, position <chr>, elias_id <chr>, full_name <chr>
```

```
save(
  nfl_depthCharts,
  file = "./data/nfl_depthCharts.RData"
)
```

4.3.12 Trades

A Data Dictionary for game schedules data is located at the following link:
https://github.com/nflverse/nfldata/blob/master/DATASETS.md#trades

```
nfl_trades_raw <- read_csv("https://raw.githubusercontent.com/leesharpe/nfldata/master/data/trades
```

```
save(
  nfl_trades_raw,
  file = "./data/nfl_trades_raw.RData"
)
```

The `nfl_trades` object is in `trade-player` form. That is, each row should be uniquely identified by the combination of `trade_id` and `pfr_id`.

```
nfl_trades <- nfl_trades_raw
```

Let's check for duplicate `trade-player` instances:

```
nfl_trades %>%
  filter(!is.na(pfr_id)) %>%
  group_by(trade_id, pfr_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 11
# Groups:   trade_id, pfr_id [0]
# i 11 variables: trade_id <dbl>, season <dbl>, trade_date <date>, gave <chr>,
#   received <chr>, pick_season <dbl>, pick_round <dbl>, pick_number <dbl>,
#   conditional <dbl>, pfr_id <chr>, pfr_name <chr>
```

```
save(
  nfl_trades,
  file = "./data/nfl_trades.RData"
)
```

4.3.13 Play-By-Play Data

To download play-by-play data from prior weeks and seasons, we can use the `load_pbp()` function of the `nflreadr` package. We add a progress bar using the `with_progress()` function from the `progressr` package because it takes a while to run. A Data Dictionary for the play-by-play data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_pbp.html

> **Note 2: Downloading play-by-play data**
>
> Note: the following code takes a while to run.

```
nfl_pbp_raw <- progressr::with_progress(
  nflreadr::load_pbp(seasons = TRUE))
```

```
save(
  nfl_pbp_raw,
  file = "./data/nfl_pbp_raw.RData"
)
```

The `nfl_pbp` object is in `game-drive-play` form. That is, each row should be uniquely identified by the combination of `game_id`, `drive`, `play_id`. Let's rearrange the data accordingly:

```
nfl_pbp <- nfl_pbp_raw %>%
  select(game_id, drive, play_id, everything()) %>%
  arrange(game_id, drive, play_id)
```

Let's check for duplicate `game-drive-play` instances:

```
nfl_pbp %>%
  group_by(game_id, drive, play_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 6 x 372
# Groups:   game_id, drive, play_id [3]
  game_id      drive play_id old_game_id home_team away_team season_type  week
  <chr>        <dbl>   <dbl> <chr>      <chr>    <chr>    <chr>      <int>
1 2000_03_PIT_C~   18    2767 2000091708 CLE      PIT      REG          3
2 2000_03_PIT_C~   18    2767 2000091708 CLE      PIT      REG          3
3 2000_03_PIT_C~   18    2768 2000091708 CLE      PIT      REG          3
4 2000_03_PIT_C~   18    2768 2000091708 CLE      PIT      REG          3
5 2000_06_WAS_P~   12    1825 2000100811 PHI      WAS      REG          6
6 2000_06_WAS_P~   12    1825 2000100811 PHI      WAS      REG          6
# i 364 more variables: posteam <chr>, posteam_type <chr>, defteam <chr>,
#   side_of_field <chr>, yardline_100 <dbl>, game_date <chr>,
#   quarter_seconds_remaining <dbl>, half_seconds_remaining <dbl>,
#  game_seconds_remaining <dbl>, game_half <chr>, quarter_end <dbl>, sp <dbl>,
#   qtr <dbl>, down <dbl>, goal_to_go <dbl>, time <chr>, yrdln <chr>,
#   ydstogo <dbl>, ydsnet <dbl>, desc <chr>, play_type <chr>,
#  yards_gained <dbl>, shotgun <dbl>, no_huddle <dbl>, qb_dropback <dbl>, ...
```

```
save(
  nfl_pbp,
  file = "./data/nfl_pbp.RData"
)
```

4.3.14 4th Down Data

> **Note 3: Downloading 4th down data**
>
> Note: the following code takes a while to run.

```
nfl_4thdown_raw <- nfl4th::load_4th_pbp(
  seasons = 2014:nflreadr::most_recent_season())
```

```
save(
  nfl_4thdown_raw,
  file = "./data/nfl_4thdown_raw.RData"
)
```

The `nfl_4thdown` object is in `game-drive-play` form. That is, each row should be uniquely identified by the combination of `game_id`, `drive`, `play_id`. Let's rearrange the data accordingly:

```
nfl_4thdown <- nfl_4thdown_raw %>%
  select(game_id, drive, play_id, everything()) %>%
  arrange(game_id, drive, play_id)
```

Let's check for duplicate `game-drive-play` instances:

```
nfl_4thdown %>%
  group_by(game_id, drive, play_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 383
# Groups:   game_id, drive, play_id [0]
# i 383 variables: game_id <chr>, drive <dbl>, play_id <dbl>,
#  old_game_id <chr>, home_team <chr>, away_team <chr>, season_type <chr>,
#   week <int>, posteam <chr>, posteam_type <chr>, defteam <chr>,
#   side_of_field <chr>, yardline_100 <dbl>, game_date <chr>,
#   quarter_seconds_remaining <dbl>, half_seconds_remaining <dbl>,
#  game_seconds_remaining <dbl>, game_half <chr>, quarter_end <dbl>, sp <dbl>,
#  qtr <dbl>, down <dbl>, goal_to_go <int>, time <chr>, yrdln <chr>, ...
```

```
save(
  nfl_4thdown,
  file = "./data/nfl_4thdown.RData"
)
```

4.3.15 Participation

A Data Dictionary for the participation data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_participation.html

```
nfl_participation_raw <- progressr::with_progress(
  nflreadr::load_participation(
    seasons = TRUE,
    include_pbp = TRUE))
```

```
save(
  nfl_participation_raw,
  file = "./data/nfl_participation_raw.RData"
)
```

The `nfl_participation` object is in `game-drive-play` form. That is, each row should be uniquely identified by the combination of `nflverse_game_id`, `drive`, `play_id`. Let's rearrange the data accordingly:

```
nfl_participation <- nfl_participation_raw %>%
  select(nflverse_game_id, drive, play_id, everything()) %>%
  arrange(nflverse_game_id, drive, play_id)
```

Let's check for duplicate `game-drive-play` instances:

```
nfl_participation %>%
  group_by(nflverse_game_id, drive, play_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 389
# Groups:   nflverse_game_id, drive, play_id [0]
# i 389 variables: nflverse_game_id <chr>, drive <dbl>, play_id <int>,
#  possession_team <chr>, offense_formation <chr>, offense_personnel <chr>,
#   defenders_in_box <int>, defense_personnel <chr>,
#  number_of_pass_rushers <int>, players_on_play <chr>, offense_players <chr>,
#   defense_players <chr>, n_offense <int>, n_defense <int>,
#  ngs_air_yards <dbl>, time_to_throw <dbl>, was_pressure <lgl>, route <chr>,
#   defense_man_zone_type <chr>, defense_coverage_type <chr>, ...
```

```
save(
  nfl_participation,
  file = "./data/nfl_participation.RData"
)
```

4.3.16 Historical Weekly Actual Player Statistics

We can download historical week-by-week actual player statistics using the `load_player_stats()` function from the `nflreadr` package. A Data Dictionary for statistics for offensive players is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_player_stats.html. A Data Dictionary for statistics for defensive players is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_player_stats_def.html.

```
nfl_actualStats_offense_weekly_raw <- progressr::with_progress(
  nflreadr::load_player_stats(
    seasons = TRUE,
    stat_type = "offense"))

nfl_actualStats_defense_weekly_raw <- progressr::with_progress(
  nflreadr::load_player_stats(
    seasons = TRUE,
    stat_type = "defense"))

nfl_actualStats_kicking_weekly_raw <- progressr::with_progress(
  nflreadr::load_player_stats(
    seasons = TRUE,
    stat_type = "kicking"))
```

```
save(
  nfl_actualStats_offense_weekly_raw, nfl_actualStats_defense_weekly_raw, nfl_actualStats_kicking_
  file = "./data/nfl_actualStats_weekly_raw.RData"
)
```

The `nfl_actualStats_weekly` objects are in `player-season-week` form. That is, each row should be uniquely identified by the combination of `player_id`, `season`, and `week`. Let's rearrange the data accordingly:

```
nfl_actualStats_offense_weekly <- nfl_actualStats_offense_weekly_raw %>%
  rename(team = recent_team) %>%
  select(player_id, season, week, everything()) %>%
  arrange(player_display_name, player_id, season, week)
```

```
nfl_actualStats_defense_weekly <- nfl_actualStats_defense_weekly_raw %>%
  select(player_id, season, week, everything()) %>%
  arrange(player_display_name, player_id, season, week)

nfl_actualStats_kicking_weekly <- nfl_actualStats_kicking_weekly_raw %>%
  select(player_id, season, week, everything()) %>%
  arrange(player_display_name, player_id, season, week)
```

Let's check for duplicate `player-season-week` instances:

```
nfl_actualStats_offense_weekly %>%
  group_by(player_id, season, week) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 53
# Groups:   player_id, season, week [0]
# i 53 variables: player_id <chr>, season <int>, week <int>, player_name <chr>,
#   player_display_name <chr>, position <chr>, position_group <chr>,
#  headshot_url <chr>, team <chr>, season_type <chr>, opponent_team <chr>,
#  completions <int>, attempts <int>, passing_yards <dbl>, passing_tds <int>,
#  interceptions <dbl>, sacks <dbl>, sack_yards <dbl>, sack_fumbles <int>,
#   sack_fumbles_lost <int>, passing_air_yards <dbl>,
#   passing_yards_after_catch <dbl>, passing_first_downs <dbl>, ...
```

```
nfl_actualStats_defense_weekly %>%
  group_by(player_id, season, week) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 6 x 32
# Groups:   player_id, season, week [3]
 player_id  season  week season_type player_name player_display_name position
  <chr>      <int> <int> <chr>       <chr>       <chr>              <chr>
1 00-0035676   2021     1 REG         A.Brown     A.J. Brown         WR
2 00-0035676   2021     1 REG         A.Brown     A.J. Brown         WR
3 00-0023459   2014    19 POST        A.Rodgers   Aaron Rodgers      QB
4 00-0023459   2014    19 POST        A.Rodgers   Aaron Rodgers      QB
5 00-0027346   2012     5 REG         <NA>        Alfonso Smith      RB
6 00-0027346   2012     5 REG         <NA>        Alfonso Smith      RB
# i 25 more variables: position_group <chr>, headshot_url <chr>, team <chr>,
#  def_tackles <int>, def_tackles_solo <int>, def_tackles_with_assist <int>,
#   def_tackle_assists <int>, def_tackles_for_loss <int>,
```

```
#   def_tackles_for_loss_yards <dbl>, def_fumbles_forced <int>,
#   def_sacks <dbl>, def_sack_yards <dbl>, def_qb_hits <dbl>,
#   def_interceptions <dbl>, def_interception_yards <dbl>,
#   def_pass_defended <dbl>, def_tds <dbl>, def_fumbles <dbl>, ...
```

```
nfl_actualStats_kicking_weekly %>%
  group_by(player_id, season, week) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 4 x 44
# Groups:   player_id, season, week [2]
 player_id  season  week season_type team  player_name player_display_name
  <chr>       <int> <int> <chr>       <chr> <chr>       <chr>
1 00-0004811   2000    11 REG         DEN   <NA>        Jason Elam
2 00-0004811   2000    11 REG         LV    <NA>        Jason Elam
3 00-0012875   2002     4 REG         PIT   <NA>        Todd Peterson
4 00-0012875   2002     4 REG         PIT   <NA>        Todd Peterson
# i 37 more variables: position <chr>, position_group <chr>,
#   headshot_url <chr>, fg_made <int>, fg_att <dbl>, fg_missed <int>,
#   fg_blocked <int>, fg_long <dbl>, fg_pct <dbl>, fg_made_0_19 <int>,
#   fg_made_20_29 <int>, fg_made_30_39 <int>, fg_made_40_49 <int>,
#   fg_made_50_59 <int>, fg_made_60_ <int>, fg_missed_0_19 <int>,
#  fg_missed_20_29 <int>, fg_missed_30_39 <int>, fg_missed_40_49 <int>,
#   fg_missed_50_59 <int>, fg_missed_60_ <int>, fg_made_list <chr>, ...
```

```
save(
  nfl_actualStats_offense_weekly, nfl_actualStats_defense_weekly, nfl_actualStats_kicking_weekly,
  file = "./data/nfl_actualStats_weekly.RData"
)
```

4.3.17 Injuries

A Data Dictionary for injury data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_injuries.html

```
nfl_injuries_raw <- progressr::with_progress(
  nflreadr::load_injuries(seasons = TRUE))
```

```
save(
  nfl_injuries_raw,
  file = "./data/nfl_injuries_raw.RData"
)
```

The `nfl_injuries` object is in `player-season-week` form. That is, each row should be uniquely identified by the combination of `gsis_id`, `season`, and `week`. Let's rearrange the data accordingly:

```
nfl_injuries <- nfl_injuries_raw %>%
  select(gsis_id, season, week, everything()) %>%
  arrange(full_name, gsis_id, season, week)
```

Let's check for duplicate `player-season-week` instances:

```
nfl_injuries %>%
  group_by(gsis_id, season, week) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 4 x 16
# Groups:   gsis_id, season, week [2]
  gsis_id   season  week game_type team  position full_name first_name last_name
  <chr>      <dbl> <dbl> <chr>    <chr> <chr>    <chr>     <chr>      <chr>
1 00-00332~   2022     7 REG      CAR   RB       Christia~ Christian  McCaffrey
2 00-00332~   2022     7 REG      SF    RB       Christia~ Christian  McCaffrey
3 00-00262~   2017    10 REG      GB    TE       Martellu~ Martellus  Bennett
4 00-00262~   2017    10 REG      NE    TE       Martellu~ Martellus  Bennett
# i 7 more variables: report_primary_injury <chr>,
#   report_secondary_injury <chr>, report_status <chr>,
#   practice_primary_injury <chr>, practice_secondary_injury <chr>,
#   practice_status <chr>, date_modified <dttm>
```

```
save(
  nfl_injuries,
  file = "./data/nfl_injuries.RData"
)
```

4.3.18 Snap Counts

A Data Dictionary for snap counts data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_snap_counts.html

```
nfl_snapCounts_raw <- progressr::with_progress(
  nflreadr::load_snap_counts(seasons = TRUE))
```

```
save(
  nfl_snapCounts_raw,
  file = "./data/nfl_snapCounts_raw.RData"
)
```

The `nfl_snapCounts` object is in `game-player` form. That is, each row should be uniquely identified by the combination of `game_id` and `pfr_player_id`. Let's rearrange the data accordingly:

```
nfl_snapCounts <- nfl_snapCounts_raw %>%
  select(game_id, pfr_player_id, everything()) %>%
  arrange(game_id, pfr_player_id)
```

Let's check for duplicate `game` instances:

```
nfl_snapCounts %>%
  group_by(game_id, pfr_player_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 16
# Groups:   game_id, pfr_player_id [0]
# i 16 variables: game_id <chr>, pfr_player_id <chr>, pfr_game_id <chr>,
#  season <int>, game_type <chr>, week <int>, player <chr>, position <chr>,
#   team <chr>, opponent <chr>, offense_snaps <dbl>, offense_pct <dbl>,
#   defense_snaps <dbl>, defense_pct <dbl>, st_snaps <dbl>, st_pct <dbl>
```

```
save(
  nfl_snapCounts,
  file = "./data/nfl_snapCounts.RData"
)
```

4.3.19 ESPN QBR

A Data Dictionary for ESPN QBR data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_espn_qbr.html

```
nfl_espnQBR_seasonal_raw <- progressr::with_progress(
  nflreadr::load_espn_qbr(
    seasons = TRUE,
    summary_type = c("season")))
```

```
nfl_espnQBR_weekly_raw <- progressr::with_progress(
  nflreadr::load_espn_qbr(
    seasons = TRUE,
    summary_type = c("week")))
```

```
save(
  nfl_espnQBR_seasonal_raw,
  file = "./data/nfl_espnQBR_seasonal_raw.RData"
)

save(
  nfl_espnQBR_weekly_raw,
  file = "./data/nfl_espnQBR_weekly_raw.RData"
)
```

The `nfl_espnQBR_seasonal` object is in `player-season-season type` form, where `season type` refers to regular season versus postseason. That is, each row should be uniquely identified by the combination of `player_id`, `season`, and `season_type`. Let's rearrange the data accordingly:

```
nfl_espnQBR_seasonal <- nfl_espnQBR_seasonal_raw %>%
  select(player_id, season, season_type, everything()) %>%
  arrange(name_display, player_id, season, season_type)
```

Let's check for duplicate `player-season-team` instances:

```
nfl_espnQBR_seasonal %>%
  group_by(player_id, season, season_type) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 23
# Groups:   player_id, season, season_type [0]
# i 23 variables: player_id <chr>, season <int>, season_type <chr>,
#   game_week <chr>, team_abb <chr>, name_short <chr>, rank <dbl>,
#   qbr_total <dbl>, pts_added <dbl>, qb_plays <dbl>, epa_total <dbl>,
#   pass <dbl>, run <dbl>, exp_sack <dbl>, penalty <dbl>, qbr_raw <dbl>,
#   sack <dbl>, name_first <chr>, name_last <chr>, name_display <chr>,
#   headshot_href <chr>, team <chr>, qualified <lgl>
```

The `nfl_espnQBR_weekly` object is in both `game-player` form and `player-season-season type-week` form, where `season type` refers to regular season versus postseason. That is, each row should be uniquely identified by the

combination of `gsis_id`, `season`, and `week` or by the combination of `player_id`, `season`, `season_type`, and `week_num`. Let's rearrange the data accordingly:

```
nfl_espnQBR_weekly <- nfl_espnQBR_weekly_raw %>%
  select(player_id, season, season_type, week_num, everything()) %>%
  arrange(name_display, player_id, season, season_type, week_num)
```

Let's check for duplicate `game-player` or `player-season-season type-week` instances:

```
nfl_espnQBR_weekly %>%
  arrange(game_id, player_id) %>%
  group_by(game_id, player_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 30
# Groups:   game_id, player_id [0]
# i 30 variables: player_id <chr>, season <int>, season_type <chr>,
#   week_num <int>, game_id <chr>, game_week <int>, week_text <chr>,
#   team_abb <chr>, name_short <chr>, rank <dbl>, qbr_total <dbl>,
#  pts_added <dbl>, qb_plays <dbl>, epa_total <dbl>, pass <dbl>, run <dbl>,
#  exp_sack <dbl>, penalty <dbl>, qbr_raw <dbl>, sack <dbl>, name_first <chr>,
#  name_last <chr>, name_display <chr>, headshot_href <chr>, team <chr>,
#   opp_id <chr>, opp_abb <chr>, opp_team <chr>, opp_name <chr>, ...
```

```
nfl_espnQBR_weekly %>%
  arrange(player_id, season, season_type, week_num) %>%
  group_by(player_id, season, season_type, week_num) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 30
# Groups:   player_id, season, season_type, week_num [0]
# i 30 variables: player_id <chr>, season <int>, season_type <chr>,
#   week_num <int>, game_id <chr>, game_week <int>, week_text <chr>,
#   team_abb <chr>, name_short <chr>, rank <dbl>, qbr_total <dbl>,
#  pts_added <dbl>, qb_plays <dbl>, epa_total <dbl>, pass <dbl>, run <dbl>,
#  exp_sack <dbl>, penalty <dbl>, qbr_raw <dbl>, sack <dbl>, name_first <chr>,
#  name_last <chr>, name_display <chr>, headshot_href <chr>, team <chr>,
#   opp_id <chr>, opp_abb <chr>, opp_team <chr>, opp_name <chr>, ...
```

```
save(
  nfl_espnQBR_seasonal,
  file = "./data/nfl_espnQBR_seasonal.RData"
)

save(
  nfl_espnQBR_weekly,
  file = "./data/nfl_espnQBR_weekly.RData"
)
```

4.3.20 NFL Next Gen Stats

A Data Dictionary for NFL Next Gen Stats data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_nextgen_stats.html

```
nfl_nextGenStats_pass_weekly_raw <- progressr::with_progress(
  nflreadr::load_nextgen_stats(
    seasons = TRUE,
    stat_type = c("passing")))

nfl_nextGenStats_rush_weekly_raw <- progressr::with_progress(
  nflreadr::load_nextgen_stats(
    seasons = TRUE,
    stat_type = c("rushing")))

nfl_nextGenStats_rec_weekly_raw <- progressr::with_progress(
  nflreadr::load_nextgen_stats(
    seasons = TRUE,
    stat_type = c("receiving")))

nfl_nextGenStats_weekly_raw <- bind_rows(
  nfl_nextGenStats_pass_weekly_raw,
  nfl_nextGenStats_rush_weekly_raw,
  nfl_nextGenStats_rec_weekly_raw
)
```

```
save(
  nfl_nextGenStats_weekly_raw,
  file = "./data/nfl_nextGenStats_weekly_raw.RData"
)
```

The `nfl_nextGenStats_weekly` object is in `player-season-season type-week` form, where `season type` refers to regular season versus postseason. That is,

each row should be uniquely identified by the combination of `player_gsis_id`, `season`, `season_type`, and `week`. Let's rearrange the data accordingly:

```
nfl_nextGenStats_weekly <- nfl_nextGenStats_weekly_raw %>%
  select(player_gsis_id, season, season_type, week, everything()) %>%
  arrange(player_display_name, player_gsis_id, season, season_type)
```

Let's check for duplicate `player-season-season type-week` instances:

```
nfl_nextGenStats_weekly %>%
  group_by(player_gsis_id, season, season_type, week) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 51
# Groups:   player_gsis_id, season, season_type, week [0]
# i 51 variables: player_gsis_id <chr>, season <int>, season_type <chr>,
#   week <int>, player_display_name <chr>, player_position <chr>,
#  team_abbr <chr>, avg_time_to_throw <dbl>, avg_completed_air_yards <dbl>,
#   avg_intended_air_yards <dbl>, avg_air_yards_differential <dbl>,
#   aggressiveness <dbl>, max_completed_air_distance <dbl>,
#   avg_air_yards_to_sticks <dbl>, attempts <int>, pass_yards <int>,
#  pass_touchdowns <int>, interceptions <int>, passer_rating <dbl>, ...
```

```
save(
  nfl_nextGenStats_weekly,
  file = "./data/nfl_nextGenStats_weekly.RData"
)
```

4.3.21 Advanced Stats from Pro Football Reference

A Data Dictionary for Pro Football Reference passing data is located at the following links:

- https://nflreadr.nflverse.com/articles/dictionary_pfr_passing.html
- https://www.pro-football-reference.com/about/advanced_stats.htm

Advanced stats from the 2023 season are available at the following link: https://www.pro-football-reference.com/years/2023/advanced.htm

```
nfl_advancedStatsPFR_pass_seasonal_raw <- progressr::with_progress(
  nflreadr::load_pfr_advstats(
    seasons = TRUE,
    stat_type = c("pass"),
    summary_level = c("season")))

nfl_advancedStatsPFR_pass_weekly_raw <- progressr::with_progress(
  nflreadr::load_pfr_advstats(
    seasons = TRUE,
    stat_type = c("pass"),
    summary_level = c("week")))

nfl_advancedStatsPFR_rush_seasonal_raw <- progressr::with_progress(
  nflreadr::load_pfr_advstats(
    seasons = TRUE,
    stat_type = c("rush"),
    summary_level = c("season")))

nfl_advancedStatsPFR_rush_weekly_raw <- progressr::with_progress(
  nflreadr::load_pfr_advstats(
    seasons = TRUE,
    stat_type = c("rush"),
    summary_level = c("week")))

nfl_advancedStatsPFR_rec_seasonal_raw <- progressr::with_progress(
  nflreadr::load_pfr_advstats(
    seasons = TRUE,
    stat_type = c("rec"),
    summary_level = c("season")))

nfl_advancedStatsPFR_rec_weekly_raw <- progressr::with_progress(
  nflreadr::load_pfr_advstats(
    seasons = TRUE,
    stat_type = c("rec"),
    summary_level = c("week")))

nfl_advancedStatsPFR_def_seasonal_raw <- progressr::with_progress(
  nflreadr::load_pfr_advstats(
    seasons = TRUE,
    stat_type = c("def"),
    summary_level = c("season")))

nfl_advancedStatsPFR_def_weekly_raw <- progressr::with_progress(
  nflreadr::load_pfr_advstats(
```

```
    seasons = TRUE,
    stat_type = c("def"),
    summary_level = c("week")))
```

```
save(
  nfl_advancedStatsPFR_pass_seasonal_raw,
  nfl_advancedStatsPFR_rush_seasonal_raw,
  nfl_advancedStatsPFR_rec_seasonal_raw,
  nfl_advancedStatsPFR_def_seasonal_raw,
  file = "./data/nfl_advancedStatsPFR_seasonal_raw.RData"
)

save(
  nfl_advancedStatsPFR_pass_weekly_raw,
  nfl_advancedStatsPFR_rush_weekly_raw,
  nfl_advancedStatsPFR_rec_weekly_raw,
  nfl_advancedStatsPFR_def_weekly_raw,
  file = "./data/nfl_advancedStatsPFR_weekly_raw.RData"
)
```

4.3.21.1 Processing

```
# Clean up player name for merging; name variables based on which data object they're from

## Seasonal Data
nfl_advancedStatsPFR_pass_seasonal <- nfl_advancedStatsPFR_pass_seasonal_raw %>%
  mutate(
    merge_name = nflreadr::clean_player_names(player, lowercase = TRUE)
  ) %>%
  rename_with(
    ~ paste0(., ".pass"),
    -c(pfr_id, merge_name, season, team))

nfl_advancedStatsPFR_rush_seasonal <- nfl_advancedStatsPFR_rush_seasonal_raw %>%
  mutate(
    merge_name = nflreadr::clean_player_names(player, lowercase = TRUE)
  ) %>%
  rename(
    team = tm
  ) %>%
  rename_with(
    ~ paste0(., ".rush"),
    -c(pfr_id, merge_name, season, team))
```

```
nfl_advancedStatsPFR_rec_seasonal <- nfl_advancedStatsPFR_rec_seasonal_raw %>%
  mutate(
    merge_name = nflreadr::clean_player_names(player, lowercase = TRUE)
  ) %>%
  rename(
    team = tm
  ) %>%
  rename_with(
    ~ paste0(., ".rec"),
    -c(pfr_id, merge_name, season, team))

nfl_advancedStatsPFR_def_seasonal <- nfl_advancedStatsPFR_def_seasonal_raw %>%
  mutate(
    merge_name = nflreadr::clean_player_names(player, lowercase = TRUE)
  )  %>%
  rename(
    team = tm
  ) %>%
  rename_with(
    ~ paste0(., ".def"),
    -c(pfr_id, merge_name, season, team))

## Weekly Data
nfl_advancedStatsPFR_pass_weekly <- nfl_advancedStatsPFR_pass_weekly_raw %>%
  mutate(
    merge_name = nflreadr::clean_player_names(pfr_player_name, lowercase = TRUE)
  ) %>%
  rename_with(
    ~ paste0(., ".pass"),
    -c(game_id, pfr_player_id))

nfl_advancedStatsPFR_rush_weekly <- nfl_advancedStatsPFR_rush_weekly_raw %>%
  mutate(
    merge_name = nflreadr::clean_player_names(pfr_player_name, lowercase = TRUE)
  ) %>%
  rename_with(
    ~ paste0(., ".rush"),
    -c(game_id, pfr_player_id))

nfl_advancedStatsPFR_rec_weekly <- nfl_advancedStatsPFR_rec_weekly_raw %>%
  mutate(
    merge_name = nflreadr::clean_player_names(pfr_player_name, lowercase = TRUE)
  ) %>%
  rename_with(
```

```
    ~ paste0(., ".rec"),
    -c(game_id, pfr_player_id))

nfl_advancedStatsPFR_def_weekly <- nfl_advancedStatsPFR_def_weekly_raw %>%
  mutate(
    merge_name = nflreadr::clean_player_names(pfr_player_name, lowercase = TRUE)
  ) %>%
  rename_with(
    ~ paste0(., ".def"),
    -c(game_id, pfr_player_id))

## Merge across positions
nfl_advancedStatsPFR_seasonal_list <- list(
  nfl_advancedStatsPFR_pass_seasonal,
  nfl_advancedStatsPFR_rush_seasonal,
  nfl_advancedStatsPFR_rec_seasonal,
  nfl_advancedStatsPFR_def_seasonal)

nfl_advancedStatsPFR_weekly_list <- list(
  nfl_advancedStatsPFR_pass_weekly,
  nfl_advancedStatsPFR_rush_weekly,
  nfl_advancedStatsPFR_rec_weekly,
  nfl_advancedStatsPFR_def_weekly)

nfl_advancedStatsPFR_seasonalByTeam <- nfl_advancedStatsPFR_seasonal_list %>%
  purrr::reduce(
    full_join,
    by = c("pfr_id","merge_name","season","team"))

nfl_advancedStatsPFR_weekly <- nfl_advancedStatsPFR_weekly_list %>%
  purrr::reduce(
    full_join,
    by = c("game_id","pfr_player_id")) #merge_name

#nfl_advancedStatsPFR_weekly <- nfl_advancedStatsPFR_weekly_list %>%
#  purrr::reduce(
#    full_join,
#    by = c("pfr_player_id","merge_name","season","week"))

nfl_advancedStatsPFR_seasonalByTeam <- nfl_advancedStatsPFR_seasonalByTeam %>%
  mutate(
    pfr_player_name = coalesce(
      player.pass,
      player.rush,
```

```
      player.rec,
      player.def
    ),
    age = coalesce(
      #age.pass,
      age.rush,
      age.rec,
      age.def
    ),
    pos = coalesce(
      #pos.pass,
      pos.rush,
      pos.rec,
      pos.def
    ),
    g = coalesce(
      #g.pass,
      g.rush,
      g.rec,
      g.def
    ),
    gs = coalesce(
      #gs.pass,
      gs.rush,
      gs.rec,
      gs.def
    )
  ) %>%
  select(-c(
    starts_with("player."),
    starts_with("age."),
    starts_with("pos."),
    starts_with("g."),
    starts_with("gs.")))

nfl_advancedStatsPFR_weekly <- nfl_advancedStatsPFR_weekly %>%
  mutate(
    pfr_player_name = coalesce(
      pfr_player_name.pass,
      pfr_player_name.rush,
      pfr_player_name.rec,
      pfr_player_name.def
    ),
    season = coalesce(
```

```
      season.pass,
      season.rush,
      season.rec,
      season.def
    ),
    week = coalesce(
      week.pass,
      week.rush,
      week.rec,
      week.def
    ),
    team = coalesce(
      team.pass,
      team.rush,
      team.rec,
      team.def
    ),
    merge_name = coalesce(
      merge_name.pass,
      merge_name.rush,
      merge_name.rec,
      merge_name.def
    ),
    pfr_game_id = coalesce(
      pfr_game_id.pass,
      pfr_game_id.rush,
      pfr_game_id.rec,
      pfr_game_id.def
    ),
    game_type = coalesce(
      game_type.pass,
      game_type.rush,
      game_type.rec,
      game_type.def
    ),
    opponent = coalesce(
      opponent.pass,
      opponent.rush,
      opponent.rec,
      opponent.def
    )
  ) %>%
  select(-c(
    starts_with("pfr_player_name."),
    starts_with("season."),
```

```
    starts_with("week."),
    starts_with("team."),
    starts_with("merge_name."),
    starts_with("pfr_game_id."),
    starts_with("game_type."),
    starts_with("opponent."),
    ))
```

The `nfl_advancedStatsPFR_seasonalByTeam` object is in `player-season-team` form. That is, each row should be uniquely identified by the combination of `pfr_id`, `season`, and `team`. Let's rearrange the data accordingly:

```
nfl_advancedStatsPFR_seasonalByTeam <- nfl_advancedStatsPFR_seasonalByTeam %>%
  select(pfr_id, season, team, pfr_player_name, everything()) %>%
  arrange(pfr_player_name, pfr_id, season, team)
```

Let's check for duplicate `player-season-team` instances:

```
nfl_advancedStatsPFR_seasonalByTeam %>%
  group_by(pfr_id, season, team) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 82
# Groups:   pfr_id, season, team [0]
# i 82 variables: pfr_id <chr>, season <int>, team <chr>,
#  pfr_player_name <chr>, pass_attempts.pass <dbl>, throwaways.pass <dbl>,
#   spikes.pass <dbl>, drops.pass <dbl>, drop_pct.pass <dbl>,
#  bad_throws.pass <dbl>, bad_throw_pct.pass <dbl>, pocket_time.pass <dbl>,
#  times_blitzed.pass <dbl>, times_hurried.pass <dbl>, times_hit.pass <dbl>,
#   times_pressured.pass <dbl>, pressure_pct.pass <dbl>,
#   batted_balls.pass <dbl>, on_tgt_throws.pass <dbl>, ...
```

Aggregate variables within each pass/rush/rec/def object by team for seasonal data (so seasonal data are in player-season form, not player-season-team form). Depending on the variable, aggregation was performed using a sum, weighted mean (weighted by the number of games played for each team), or a recomputed percentage.

```
pfrVars <- nfl_advancedStatsPFR_seasonalByTeam %>%
  select(pass_attempts.pass:m_tkl_percent.def, g, gs) %>%
  names()
```

```
weightedAverageVars <- c(
  "pocket_time.pass",
  "ybc_att.rush","yac_att.rush",
  "ybc_r.rec","yac_r.rec","adot.rec","rat.rec",
  "yds_cmp.def","yds_tgt.def","dadot.def","m_tkl_percent.def","rat.def"
)

recomputeVars <- c(
  "drop_pct.pass", # drops.pass / pass_attempts.pass
  "bad_throw_pct.pass", # bad_throws.pass / pass_attempts.pass
  "on_tgt_pct.pass", # on_tgt_throws.pass / pass_attempts.pass
  "pressure_pct.pass", # times_pressured.pass / pass_attempts.pass
  "drop_percent.rec", # drop.rec / tgt.rec
  "rec_br.rec", # rec.rec / brk_tkl.rec
  "cmp_percent.def" # cmp.def / tgt.def
)

sumVars <- pfrVars[pfrVars %ni% c(
  weightedAverageVars, recomputeVars,
  "merge_name", "loaded.pass", "loaded.rush", "loaded.rec", "loaded.def")]

nfl_advancedStatsPFR_seasonal <- nfl_advancedStatsPFR_seasonalByTeam %>%
  group_by(pfr_id, merge_name, season) %>%
  summarise(
    across(all_of(weightedAverageVars), ~ weighted.mean(.x, w = g, na.rm = TRUE)),
    across(all_of(sumVars), ~ sum(.x, na.rm = TRUE)),
    .groups = "drop") %>%
  mutate(
    drop_pct.pass = drops.pass / pass_attempts.pass,
    bad_throw_pct.pass = bad_throws.pass / pass_attempts.pass,
    on_tgt_pct.pass = on_tgt_throws.pass / pass_attempts.pass,
    pressure_pct.pass = times_pressured.pass / pass_attempts.pass,
    drop_percent.rec = drop.rec / tgt.rec,
    rec_br.rec = drop.rec / tgt.rec,
    cmp_percent.def = cmp.def / tgt.def
  )

# Merge with other player info
nfl_advancedStatsPFR_seasonalByTeam_1stTeam <- nfl_advancedStatsPFR_seasonalByTeam %>%
  group_by(pfr_id, season) %>%
  slice(1)

nfl_advancedStatsPFR_seasonalByTeam_1stTeam_mergeVars <- nfl_advancedStatsPFR_seasonalByTeam_1stTe
  select(pfr_id, season, team, pfr_player_name, age, pos) #, g, gs
```

```
nfl_advancedStatsPFR_seasonal <- nfl_advancedStatsPFR_seasonal %>%
  left_join(
    nfl_advancedStatsPFR_seasonalByTeam_1stTeam_mergeVars,
    by = c("pfr_id","season")
  ) %>%
  select(
    pfr_id, season, pfr_player_name, pos, age, team, g, gs,
    contains(".pass"), contains(".rush"), contains(".rec"), contains(".def"),
    everything())
```

Let's check for duplicate `player-season` instances:

```
nfl_advancedStatsPFR_seasonal %>%
  group_by(pfr_id, season) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 79
# Groups:   pfr_id, season [0]
# i 79 variables: pfr_id <chr>, season <int>, pfr_player_name <chr>, pos <chr>,
#   age <dbl>, team <chr>, g <dbl>, gs <dbl>, pocket_time.pass <dbl>,
#   pass_attempts.pass <dbl>, throwaways.pass <dbl>, spikes.pass <dbl>,
#   drops.pass <dbl>, bad_throws.pass <dbl>, times_blitzed.pass <dbl>,
#  times_hurried.pass <dbl>, times_hit.pass <dbl>, times_pressured.pass <dbl>,
#  batted_balls.pass <dbl>, on_tgt_throws.pass <dbl>, rpo_plays.pass <dbl>,
#   rpo_yards.pass <dbl>, rpo_pass_att.pass <dbl>, ...
```

The `nfl_advancedStatsPFR_weekly` object is in both `game-player` form and `player-season-week` form. That is, each row should be uniquely identified by the combination of `pfr_player_id`, `season`, and `week` or by the combination of `pfr_player_id`, `season`, `game_type`, and `week`. Let's rearrange the data accordingly:

```
nfl_advancedStatsPFR_weekly <- nfl_advancedStatsPFR_weekly %>%
  select(pfr_player_id, season, week, game_type, game_id, pfr_player_name, everything()) %>%
  arrange(pfr_player_name, pfr_player_id, season, week)
```

Let's check for duplicate `game-player` or `player-season-week` instances:

```
nfl_advancedStatsPFR_weekly %>%
  arrange(game_id, pfr_player_id) %>%
  group_by(game_id, pfr_player_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 60
# Groups:   game_id, pfr_player_id [0]
# i 60 variables: pfr_player_id <chr>, season <int>, week <int>,
#   game_type <chr>, game_id <chr>, pfr_player_name <chr>,
#   passing_drops.pass <dbl>, passing_drop_pct.pass <dbl>,
#   receiving_drop.pass <dbl>, receiving_drop_pct.pass <dbl>,
#   passing_bad_throws.pass <dbl>, passing_bad_throw_pct.pass <dbl>,
#   times_sacked.pass <dbl>, times_blitzed.pass <dbl>,
#   times_hurried.pass <dbl>, times_hit.pass <dbl>, ...
```

```r
nfl_advancedStatsPFR_weekly %>%
  arrange(pfr_player_id, season, week) %>%
  group_by(pfr_player_id, season, week) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 60
# Groups:   pfr_player_id, season, week [0]
# i 60 variables: pfr_player_id <chr>, season <int>, week <int>,
#   game_type <chr>, game_id <chr>, pfr_player_name <chr>,
#   passing_drops.pass <dbl>, passing_drop_pct.pass <dbl>,
#   receiving_drop.pass <dbl>, receiving_drop_pct.pass <dbl>,
#   passing_bad_throws.pass <dbl>, passing_bad_throw_pct.pass <dbl>,
#   times_sacked.pass <dbl>, times_blitzed.pass <dbl>,
#   times_hurried.pass <dbl>, times_hit.pass <dbl>, ...
```

Merge with `gsis_id` for merging with other datasets:

```r
# Prepare data for merging
nfl_advancedStatsPFR_weekly <- nfl_advancedStatsPFR_weekly %>%
  rename(pfr_id = pfr_player_id)

# Identify duplicates
nfl_advancedStatsPFR_seasonal %>%
  select(pfr_id, season, age) %>%
  na.omit() %>%
  unique() %>%
  group_by(pfr_id, season) %>%
  filter(n() > 1) %>%
  arrange(pfr_id, season) %>%
  head()
```

```
# A tibble: 0 x 3
# Groups:   pfr_id, season [0]
# i 3 variables: pfr_id <chr>, season <int>, age <dbl>
```

```
# Merge seasonal data with the player IDs
nfl_advancedStatsPFR_seasonal <- left_join(
  nfl_advancedStatsPFR_seasonal,
  nfl_playerIDs %>%
    filter(!is.na(pfr_id)) %>%
    filter(gsis_id != "00-0039137") %>% # drop DL Byron Young, keep OLB Byron Young
    select(pfr_id, gsis_id) %>%
    unique(),
  by = "pfr_id"
)

# Merge weekly data with the player IDs
nfl_advancedStatsPFR_weekly <- left_join(
  nfl_advancedStatsPFR_weekly,
  nfl_playerIDs %>%
    filter(!is.na(pfr_id)) %>%
    filter(gsis_id != "00-0039137") %>% # drop DL Byron Young, keep OLB Byron Young
    select(pfr_id, gsis_id) %>%
    unique(),
  by = "pfr_id"
)

# Remove distinct players who were given the same `pfr_id` (to allow merging)
nfl_advancedStatsPFR_seasonal$gsis_id[which(nfl_advancedStatsPFR_seasonal$gsis_id == "00-0035665" &
#nfl_advancedStatsPFR_weekly$gsis_id[which(nfl_advancedStatsPFR_weekly$gsis_id == "00-0035665" & n

nfl_advancedStatsPFR_seasonal$gsis_id[which(nfl_advancedStatsPFR_seasonal$gsis_id == "00-0035292" &
nfl_advancedStatsPFR_weekly$gsis_id[which(nfl_advancedStatsPFR_weekly$gsis_id == "00-0035292" & nf

nfl_advancedStatsPFR_seasonal$gsis_id[which(nfl_advancedStatsPFR_seasonal$gsis_id == "00-0033894" &
#nfl_advancedStatsPFR_weekly$gsis_id[which(nfl_advancedStatsPFR_weekly$gsis_id == "00-0033894" & n

nfl_advancedStatsPFR_seasonal$gsis_id[which(nfl_advancedStatsPFR_seasonal$gsis_id == "00-0038407" &
#nfl_advancedStatsPFR_weekly$gsis_id[which(nfl_advancedStatsPFR_weekly$gsis_id == "00-0038407" & n

nfl_advancedStatsPFR_seasonal$gsis_id[which(nfl_advancedStatsPFR_seasonal$gsis_id == "00-0037106" &
#nfl_advancedStatsPFR_weekly$gsis_id[which(nfl_advancedStatsPFR_weekly$gsis_id == "00-0037106" & n

nfl_advancedStatsPFR_seasonal$gsis_id[which(nfl_advancedStatsPFR_seasonal$gsis_id == "00-0038549" &
#nfl_advancedStatsPFR_weekly$gsis_id[which(nfl_advancedStatsPFR_weekly$gsis_id == "00-0038549" & n
```

Now, each row of the `nfl_advancedStatsPFR_seasonal` object should be uniquely identified by the combination of `gsis_id` (or `pfr_id`), and `season`. Each row of the `nfl_advancedStatsPFR_weekly` object should be uniquely identified by the combination of `gsis_id` (or `pfr_id`), `season`, and `week` or by the

combination of `gsis_id` (or `pfr_id`), `season`, `game_type`, and `week`.

Let's check again for duplicate `game-player` or `player-season-week` instances:

```
# Based on gsis_id
nfl_advancedStatsPFR_seasonal %>%
  select(gsis_id, everything()) %>%
  filter(!is.na(gsis_id)) %>%
  group_by(gsis_id, season) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 80
# Groups:   gsis_id, season [0]
# i 80 variables: gsis_id <chr>, pfr_id <chr>, season <int>,
#  pfr_player_name <chr>, pos <chr>, age <dbl>, team <chr>, g <dbl>, gs <dbl>,
#  pocket_time.pass <dbl>, pass_attempts.pass <dbl>, throwaways.pass <dbl>,
#   spikes.pass <dbl>, drops.pass <dbl>, bad_throws.pass <dbl>,
#  times_blitzed.pass <dbl>, times_hurried.pass <dbl>, times_hit.pass <dbl>,
#   times_pressured.pass <dbl>, batted_balls.pass <dbl>,
#  on_tgt_throws.pass <dbl>, rpo_plays.pass <dbl>, rpo_yards.pass <dbl>, ...
```

```
nfl_advancedStatsPFR_weekly %>%
  select(gsis_id, everything()) %>%
  filter(!is.na(gsis_id)) %>%
  arrange(game_id, gsis_id) %>%
  group_by(game_id, gsis_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 61
# Groups:   game_id, gsis_id [0]
# i 61 variables: gsis_id <chr>, pfr_id <chr>, season <int>, week <int>,
#   game_type <chr>, game_id <chr>, pfr_player_name <chr>,
#   passing_drops.pass <dbl>, passing_drop_pct.pass <dbl>,
#   receiving_drop.pass <dbl>, receiving_drop_pct.pass <dbl>,
#   passing_bad_throws.pass <dbl>, passing_bad_throw_pct.pass <dbl>,
#   times_sacked.pass <dbl>, times_blitzed.pass <dbl>,
#   times_hurried.pass <dbl>, times_hit.pass <dbl>, ...
```

```
nfl_advancedStatsPFR_weekly %>%
  select(gsis_id, everything()) %>%
  filter(!is.na(gsis_id)) %>%
  arrange(gsis_id, season, week) %>%
```

```
  group_by(gsis_id, season, week) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 61
# Groups:   gsis_id, season, week [0]
# i 61 variables: gsis_id <chr>, pfr_id <chr>, season <int>, week <int>,
#   game_type <chr>, game_id <chr>, pfr_player_name <chr>,
#   passing_drops.pass <dbl>, passing_drop_pct.pass <dbl>,
#   receiving_drop.pass <dbl>, receiving_drop_pct.pass <dbl>,
#   passing_bad_throws.pass <dbl>, passing_bad_throw_pct.pass <dbl>,
#   times_sacked.pass <dbl>, times_blitzed.pass <dbl>,
#   times_hurried.pass <dbl>, times_hit.pass <dbl>, ...
```

```
# Based on pfr_id
nfl_advancedStatsPFR_seasonal %>%
  group_by(pfr_id, season) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 80
# Groups:   pfr_id, season [0]
# i 80 variables: pfr_id <chr>, season <int>, pfr_player_name <chr>, pos <chr>,
#   age <dbl>, team <chr>, g <dbl>, gs <dbl>, pocket_time.pass <dbl>,
#   pass_attempts.pass <dbl>, throwaways.pass <dbl>, spikes.pass <dbl>,
#   drops.pass <dbl>, bad_throws.pass <dbl>, times_blitzed.pass <dbl>,
#  times_hurried.pass <dbl>, times_hit.pass <dbl>, times_pressured.pass <dbl>,
#  batted_balls.pass <dbl>, on_tgt_throws.pass <dbl>, rpo_plays.pass <dbl>,
#   rpo_yards.pass <dbl>, rpo_pass_att.pass <dbl>, ...
```

```
nfl_advancedStatsPFR_weekly %>%
  arrange(game_id, pfr_id) %>%
  group_by(game_id, pfr_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 61
# Groups:   game_id, pfr_id [0]
# i 61 variables: pfr_id <chr>, season <int>, week <int>, game_type <chr>,
#   game_id <chr>, pfr_player_name <chr>, passing_drops.pass <dbl>,
#   passing_drop_pct.pass <dbl>, receiving_drop.pass <dbl>,
#   receiving_drop_pct.pass <dbl>, passing_bad_throws.pass <dbl>,
#   passing_bad_throw_pct.pass <dbl>, times_sacked.pass <dbl>,
```

```
#  times_blitzed.pass <dbl>, times_hurried.pass <dbl>, times_hit.pass <dbl>,
#   times_pressured.pass <dbl>, times_pressured_pct.pass <dbl>, ...
```

```
nfl_advancedStatsPFR_weekly %>%
  arrange(pfr_id, season, week) %>%
  group_by(pfr_id, season, week) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 61
# Groups:   pfr_id, season, week [0]
# i 61 variables: pfr_id <chr>, season <int>, week <int>, game_type <chr>,
#   game_id <chr>, pfr_player_name <chr>, passing_drops.pass <dbl>,
#   passing_drop_pct.pass <dbl>, receiving_drop.pass <dbl>,
#   receiving_drop_pct.pass <dbl>, passing_bad_throws.pass <dbl>,
#   passing_bad_throw_pct.pass <dbl>, times_sacked.pass <dbl>,
#  times_blitzed.pass <dbl>, times_hurried.pass <dbl>, times_hit.pass <dbl>,
#   times_pressured.pass <dbl>, times_pressured_pct.pass <dbl>, ...
```

```
save(
  nfl_advancedStatsPFR_seasonal,
  file = "./data/nfl_advancedStatsPFR_seasonal.RData"
)

save(
  nfl_advancedStatsPFR_weekly,
  file = "./data/nfl_advancedStatsPFR_weekly.RData"
)
```

4.3.22 Player Contracts

A Data Dictionary for player contracts data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_contracts.html

```
nfl_playerContracts_raw <- progressr::with_progress(
  nflreadr::load_contracts())
```

```
save(
  nfl_playerContracts_raw,
  file = "./data/nfl_playerContracts_raw.RData"
)
```

The `nfl_playerContracts` object is in `player-year-team-value` form. That is, each row should be uniquely identified by the combination of `otc_id`, `year_signed`, `team`, and `value`. Let's rearrange the data accordingly:

```
nfl_playerContracts <- nfl_playerContracts_raw %>%
  select(otc_id, year_signed, team, everything()) %>%
  arrange(player, otc_id, year_signed, team)
```

Let's check for duplicate `player-year-team-value` instances:

```
nfl_playerContracts %>%
  group_by(otc_id, year_signed, team, value) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 6 x 24
# Groups:   otc_id, year_signed, team, value [3]
  otc_id year_signed team     player     position is_active years value  apy
   <int>      <int> <chr>    <chr>      <chr>    <lgl>    <int> <dbl> <dbl>
1   9880       2022 Lions    A.J. Parker  CB      FALSE        1 0.207 0.207
2   9880       2022 Lions    A.J. Parker  CB      FALSE        1 0.207 0.207
3   9880       2023 Panthers  A.J. Parker  CB      FALSE        1 0.216 0.216
4   9880       2023 Panthers  A.J. Parker  CB      FALSE        1 0.216 0.216
5   8348       2019 Cardinals A.J. Richar~ WR      FALSE        1 0.136 0.136
6   8348       2019 Cardinals A.J. Richar~ WR      FALSE        1 0.136 0.136
# i 15 more variables: guaranteed <dbl>, apy_cap_pct <dbl>,
#   inflated_value <dbl>, inflated_apy <dbl>, inflated_guaranteed <dbl>,
#   player_page <chr>, date_of_birth <chr>, height <chr>, weight <chr>,
#   college <chr>, draft_year <int>, draft_round <int>, draft_overall <int>,
#   draft_team <chr>, cols <list>
```

```
save(
  nfl_playerContracts,
  file = "./data/nfl_playerContracts.RData"
)
```

4.3.23 FTN Charting Data

A Data Dictionary for FTN Charting data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_ftn_charting.html

```
nfl_ftnCharting_raw <- progressr::with_progress(
  nflreadr::load_ftn_charting(seasons = TRUE))
```

```
save(
  nfl_ftnCharting_raw,
```

```
  file = "./data/nfl_ftnCharting_raw.RData"
)
```

The `nfl_ftnCharting` object is in `game-play` form. That is, each row should be uniquely identified by the combination of `nflverse_game_id` and `play_id`. Let's rearrange the data accordingly:

```
nfl_ftnCharting <- nfl_ftnCharting_raw %>%
  select(nflverse_game_id, nflverse_play_id, everything()) %>%
  arrange(nflverse_game_id, nflverse_play_id)
```

Let's check for duplicate `game-drive-play` instances:

```
nfl_ftnCharting %>%
  group_by(nflverse_game_id, nflverse_play_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 30
# Groups:   nflverse_game_id, nflverse_play_id [0]
# i 30 variables: nflverse_game_id <chr>, nflverse_play_id <int>,
#   ftn_game_id <int>, season <int>, week <int>, ftn_play_id <int>,
#   starting_hash <chr>, qb_location <chr>, n_offense_backfield <int>,
#   n_defense_box <int>, is_no_huddle <lgl>, is_motion <lgl>,
#   is_play_action <lgl>, is_screen_pass <lgl>, is_rpo <lgl>,
#   is_trick_play <lgl>, is_qb_out_of_pocket <lgl>,
#  is_interception_worthy <lgl>, is_throw_away <lgl>, read_thrown <chr>, ...
```

```
save(
  nfl_ftnCharting,
  file = "./data/nfl_ftnCharting.RData"
)
```

4.3.24 FantasyPros Rankings

A Data Dictionary for FantasyPros ranking data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_ff_rankings.html

```
#nfl_rankings_raw <- progressr::with_progress( # currently throws error
#  nflreadr::load_ff_rankings(type = "all"))

nfl_rankings_draft_raw <- progressr::with_progress(
```

```
  nflreadr::load_ff_rankings(type = "draft"))

nfl_rankings_weekly_raw <- progressr::with_progress(
  nflreadr::load_ff_rankings(type = "week"))
```

```
save(
  nfl_rankings_draft_raw,
  file = "./data/nfl_rankings_draft_raw.RData"
)

save(
  nfl_rankings_weekly_raw,
  file = "./data/nfl_rankings_weekly_raw.RData"
)
```

The `nfl_rankings_draft` object is in `player-page_type` form. That is, each row should be uniquely identified by the player's `id`. Let's rearrange the data accordingly:

```
nfl_rankings_draft <- nfl_rankings_draft_raw %>%
  select(id, page_type, player, pos, team, everything()) %>%
  arrange(player, id, pos, page_type)
```

Let's check for duplicate `player-page_type` instances:

```
nfl_rankings_draft %>%
  group_by(id, page_type) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 25
# Groups:   id, page_type [0]
# i 25 variables: id <chr>, page_type <chr>, player <chr>, pos <chr>,
#  team <chr>, fp_page <chr>, ecr_type <chr>, ecr <dbl>, sd <dbl>, best <dbl>,
#  worst <dbl>, sportsdata_id <chr>, player_filename <chr>, yahoo_id <chr>,
#   cbs_id <chr>, player_owned_avg <dbl>, player_owned_espn <dbl>,
#   player_owned_yahoo <dbl>, player_image_url <chr>,
#   player_square_image_url <chr>, rank_delta <dbl>, bye <dbl>,
#   mergename <chr>, scrape_date <date>, tm <chr>
```

The `nfl_rankings_weekly` object is in `player-page` form. That is, each row should be uniquely identified by `fantasypros_id` and `page`. Let's rearrange the data accordingly:

```
nfl_rankings_weekly <- nfl_rankings_weekly_raw %>%
  select(fantasypros_id, page, player_name, pos, team, everything()) %>%
  arrange(player_name, fantasypros_id, page, pos)
```

Let's check for duplicate `player-page` instances:

```
nfl_rankings_weekly %>%
  group_by(fantasypros_id, page) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 36
# Groups:   fantasypros_id, page [0]
# i 36 variables: fantasypros_id <chr>, page <chr>, player_name <chr>,
#  pos <chr>, team <chr>, page_pos <chr>, scrape_date <date>, rank <dbl>,
#   ecr <dbl>, sd <dbl>, best <dbl>, worst <dbl>, sportradar_id <chr>,
#   yahoo_id <chr>, cbs_id <chr>, player_positions <chr>,
#   player_short_name <chr>, player_eligibility <chr>,
#  player_yahoo_positions <chr>, player_page_url <chr>, player_filename <chr>,
#   player_square_image_url <chr>, player_image_url <chr>, ...
```

```
save(
  nfl_rankings_draft,
  file = "./data/nfl_rankings_draft.RData"
)

save(
  nfl_rankings_weekly,
  file = "./data/nfl_rankings_weekly.RData"
)
```

4.3.25 Expected Fantasy Points

A Data Dictionary for expected fantasy points data is located at the following link: https://nflreadr.nflverse.com/articles/dictionary_ff_opportunity.html

```
nfl_expectedFantasyPoints_weekly_raw <- progressr::with_progress(
  nflreadr::load_ff_opportunity(
    seasons = TRUE,
    stat_type = "weekly",
    model_version = "latest"
  ))
```

```
nfl_expectedFantasyPoints_pass_raw <- progressr::with_progress(
  nflreadr::load_ff_opportunity(
    seasons = TRUE,
    stat_type = "pbp_pass",
    model_version = "latest"
  ))

nfl_expectedFantasyPoints_rush_raw <- progressr::with_progress(
  nflreadr::load_ff_opportunity(
    seasons = TRUE,
    stat_type = "pbp_rush",
    model_version = "latest"
  ))
```

```
nfl_expectedFantasyPoints_weekly_raw <- nflreadr::load_ff_opportunity(
  seasons = TRUE,
  stat_type = "weekly",
  model_version = "latest"
  )

nfl_expectedFantasyPoints_pass_raw <- nflreadr::load_ff_opportunity(
  seasons = TRUE,
  stat_type = "pbp_pass",
  model_version = "latest"
  )

nfl_expectedFantasyPoints_rush_raw <- nflreadr::load_ff_opportunity(
  seasons = TRUE,
  stat_type = "pbp_rush",
  model_version = "latest"
  )
```

```
save(
  nfl_expectedFantasyPoints_weekly_raw,
  file = "./data/nfl_expectedFantasyPoints_weekly_raw.RData"
)

save(
  nfl_expectedFantasyPoints_pass_raw,
  nfl_expectedFantasyPoints_rush_raw,
  file = "./data/nfl_expectedFantasyPoints_pbp_raw.RData"
)
```

```
nfl_expectedFantasyPoints_pbp_list <- list(
  nfl_expectedFantasyPoints_pass_raw,
  nfl_expectedFantasyPoints_rush_raw)

nfl_expectedFantasyPoints_pbp <- full_join(
  nfl_expectedFantasyPoints_pass_raw,
  nfl_expectedFantasyPoints_rush_raw,
  by = c("game_id","fixed_drive","play_id"),
  suffix = c(".pass", ".rush")
)
```

The `nfl_expectedFantasyPoints_weekly` object is in `game-player` form and in `player-season-week` form. That is, each row should be uniquely identified by the combination of `game_id` and `player_id`. Each row should also be uniquely identified by the combination of `player_id`, `season`, and `week`. Let's rearrange the data accordingly:

```
nfl_expectedFantasyPoints_weekly <- nfl_expectedFantasyPoints_weekly_raw %>%
  select(player_id, season, week, game_id, full_name, posteam, position, everything()) %>%
  arrange(full_name, player_id, season, week)
```

Let's check for duplicate `game-player` instances and `player-season-week` instances:

```
nfl_expectedFantasyPoints_weekly %>%
  group_by(game_id, player_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 6 x 159
# Groups:   game_id, player_id [3]
  player_id  season  week game_id      full_name posteam position pass_attempt
  <chr>     <chr>  <dbl> <chr>        <chr>    <chr>   <chr>         <dbl>
1 00-0028079 2013     3 2013_03_JAX_S~ Chris Wh~ JAX    OLB             0
2 00-0028079 2013    17 2013_17_BUF_NE Chris Wh~ NE     OLB             0
3 00-0028079 2013     3 2013_03_JAX_S~ D.J. Wil~ JAX    TE              0
4 00-0028079 2013    17 2013_17_BUF_NE D.J. Wil~ NE     TE              0
5 <NA>      2006     1 2006_01_ATL_C~ <NA>     ATL    <NA>             0
6 <NA>      2006     1 2006_01_ATL_C~ <NA>     CAR    <NA>             0
# i 151 more variables: rec_attempt <dbl>, rush_attempt <dbl>,
#   pass_air_yards <dbl>, rec_air_yards <dbl>, pass_completions <dbl>,
#   receptions <dbl>, pass_completions_exp <dbl>, receptions_exp <dbl>,
#  pass_yards_gained <dbl>, rec_yards_gained <dbl>, rush_yards_gained <dbl>,
#   pass_yards_gained_exp <dbl>, rec_yards_gained_exp <dbl>,
```

```
#  rush_yards_gained_exp <dbl>, pass_touchdown <dbl>, rec_touchdown <dbl>,
#   rush_touchdown <dbl>, pass_touchdown_exp <dbl>, ...
```

```
nfl_expectedFantasyPoints_weekly %>%
  group_by(player_id, season, week) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 6 x 159
# Groups:   player_id, season, week [3]
  player_id  season  week game_id      full_name posteam position pass_attempt
  <chr>     <chr>  <dbl> <chr>        <chr>    <chr>   <chr>         <dbl>
1 00-0028079 2013     3 2013_03_JAX_S~ Chris Wh~ JAX    OLB             0
2 00-0028079 2013    17 2013_17_BUF_NE Chris Wh~ NE     OLB             0
3 00-0028079 2013     3 2013_03_JAX_S~ D.J. Wil~ JAX    TE              0
4 00-0028079 2013    17 2013_17_BUF_NE D.J. Wil~ NE     TE              0
5 <NA>      2006     1 2006_01_ATL_C~ <NA>     ATL    <NA>             0
6 <NA>      2006     1 2006_01_ATL_C~ <NA>     CAR    <NA>             0
# i 151 more variables: rec_attempt <dbl>, rush_attempt <dbl>,
#   pass_air_yards <dbl>, rec_air_yards <dbl>, pass_completions <dbl>,
#   receptions <dbl>, pass_completions_exp <dbl>, receptions_exp <dbl>,
#  pass_yards_gained <dbl>, rec_yards_gained <dbl>, rush_yards_gained <dbl>,
#   pass_yards_gained_exp <dbl>, rec_yards_gained_exp <dbl>,
#  rush_yards_gained_exp <dbl>, pass_touchdown <dbl>, rec_touchdown <dbl>,
#   rush_touchdown <dbl>, pass_touchdown_exp <dbl>, ...
```

4.3.25.1 Processing

Let's do some data cleanup:

```
# Drop players with missing values for player_id
nfl_expectedFantasyPoints_weekly <- nfl_expectedFantasyPoints_weekly %>%
  filter(!is.na(player_id))
```

Let's check again for duplicate `game-player` instances and `season-week-player` instances:

```
nfl_expectedFantasyPoints_weekly %>%
  group_by(game_id, player_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 4 x 159
```

```
# Groups:   game_id, player_id [2]
  player_id  season  week game_id      full_name posteam position pass_attempt
  <chr>     <chr>  <dbl> <chr>        <chr>    <chr>   <chr>         <dbl>
1 00-0028079 2013     3 2013_03_JAX_S~ Chris Wh~ JAX    OLB             0
2 00-0028079 2013    17 2013_17_BUF_NE Chris Wh~ NE     OLB             0
3 00-0028079 2013     3 2013_03_JAX_S~ D.J. Wil~ JAX    TE              0
4 00-0028079 2013    17 2013_17_BUF_NE D.J. Wil~ NE     TE              0
# i 151 more variables: rec_attempt <dbl>, rush_attempt <dbl>,
#   pass_air_yards <dbl>, rec_air_yards <dbl>, pass_completions <dbl>,
#   receptions <dbl>, pass_completions_exp <dbl>, receptions_exp <dbl>,
#   pass_yards_gained <dbl>, rec_yards_gained <dbl>, rush_yards_gained <dbl>,
#   pass_yards_gained_exp <dbl>, rec_yards_gained_exp <dbl>,
#   rush_yards_gained_exp <dbl>, pass_touchdown <dbl>, rec_touchdown <dbl>,
#   rush_touchdown <dbl>, pass_touchdown_exp <dbl>, ...
```

```
nfl_expectedFantasyPoints_weekly %>%
  group_by(player_id, season, week) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 4 x 159
# Groups:   player_id, season, week [2]
  player_id  season  week game_id      full_name posteam position pass_attempt
  <chr>     <chr>  <dbl> <chr>        <chr>    <chr>   <chr>         <dbl>
1 00-0028079 2013     3 2013_03_JAX_S~ Chris Wh~ JAX    OLB             0
2 00-0028079 2013    17 2013_17_BUF_NE Chris Wh~ NE     OLB             0
3 00-0028079 2013     3 2013_03_JAX_S~ D.J. Wil~ JAX    TE              0
4 00-0028079 2013    17 2013_17_BUF_NE D.J. Wil~ NE     TE              0
# i 151 more variables: rec_attempt <dbl>, rush_attempt <dbl>,
#   pass_air_yards <dbl>, rec_air_yards <dbl>, pass_completions <dbl>,
#   receptions <dbl>, pass_completions_exp <dbl>, receptions_exp <dbl>,
#   pass_yards_gained <dbl>, rec_yards_gained <dbl>, rush_yards_gained <dbl>,
#   pass_yards_gained_exp <dbl>, rec_yards_gained_exp <dbl>,
#   rush_yards_gained_exp <dbl>, pass_touchdown <dbl>, rec_touchdown <dbl>,
#   rush_touchdown <dbl>, pass_touchdown_exp <dbl>, ...
```

The `nfl_expectedFantasyPoints_pbp` object is in `game-drive-play` form. That is, each row should be uniquely identified by the combination of `game_id`, `fixed_drive`, and `play_id`. Let's rearrange the data accordingly:

```
nfl_expectedFantasyPoints_pbp <- nfl_expectedFantasyPoints_pbp %>%
  select(game_id, fixed_drive, play_id, everything()) %>%
  arrange(game_id, fixed_drive, play_id)
```

Let's check for duplicate `game-player` instances and `season-week-player` instances:

```
nfl_expectedFantasyPoints_pbp %>%
  group_by(game_id, fixed_drive, play_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 4 x 104
# Groups:   game_id, fixed_drive, play_id [2]
 game_id      fixed_drive play_id desc.pass passer_player_id passer_full_name
  <chr>                <dbl>   <dbl> <chr>     <chr>            <chr>
1 2013_03_JAX_S~         27    4000 (3:49) 7~ 00-0026197      Chad Henne
2 2013_03_JAX_S~         27    4000 (3:49) 7~ 00-0026197      Chad Henne
3 2013_17_BUF_NE         15    2962 (:35) 12~ 00-0019596       Tom Brady
4 2013_17_BUF_NE         15    2962 (:35) 12~ 00-0019596       Tom Brady
# i 98 more variables: passer_position <chr>, receiver_player_id <chr>,
#  receiver_full_name <chr>, receiver_position <chr>, posteam.pass <chr>,
#   two_point_attempt.pass <dbl>, two_point_converted.pass <dbl>,
#   pass_attempt <dbl>, receiving_yards <dbl>, first_down_pass <dbl>,
#   fumble_lost.pass <dbl>, season.pass <int>, week.pass <int>,
#   complete_pass <fct>, yards_after_catch <dbl>, pass_touchdown <fct>,
#  first_down.pass <fct>, interception <fct>, relative_to_endzone <dbl>, ...
```

```
save(
  nfl_expectedFantasyPoints_weekly,
  file = "./data/nfl_expectedFantasyPoints_weekly.RData"
)

save(
  nfl_expectedFantasyPoints_pbp,
  file = "./data/nfl_expectedFantasyPoints_pbp.RData"
)
```

4.3.26 Fantasy Football Projections

4.3.26.1 Download Players' Projections

> **Note 4: Downloading players' seasonal projections**
>
> Note: the following code takes a while to run.

```
players_projections_seasonal_raw <- ffanalytics::scrape_data(
  season = NULL, # NULL grabs the current season
  week = 0) # NULL grabs seasonal projections; NULL grabs the current week
```

```
save(
  players_projections_seasonal_raw,
  file = "./data/players_projections_seasonal_raw.RData"
)
```

The data file is saved in the project repository and can be loaded using the following command:

```
load(file = "./data/players_projections_seasonal_raw.RData")
```

The `players_projections_seasonal_raw` object is in `player-position-projection source` form. That is, each row should be uniquely identified by the combination of `id`, `pos` and `data_src`. Each row should also be uniquely identified by the combination of `player`, `pos`, and `data_src`.

Let's check for duplicate `player-position-projection source` instances:

```
players_projections_seasonal_raw %>%
  bind_rows() %>%
  select(id, pos, data_src, everything()) %>%
  arrange(player, id, pos, data_src) %>%
  group_by(id, pos, data_src) %>%
  filter(n() > 1, !is.na(id)) %>%
  head()
```

```
# A tibble: 0 x 130
# Groups:   id, pos, data_src [0]
# i 130 variables: id <chr>, pos <chr>, data_src <chr>, player <chr>,
#  team <chr>, games <int>, pass_att <dbl>, pass_comp <dbl>, pass_yds <dbl>,
#   pass_yds_g <dbl>, pass_tds <dbl>, pass_int <dbl>, pass_rate <dbl>,
#   rush_att <dbl>, rush_yds <dbl>, rush_avg <dbl>, rush_tds <dbl>,
#   fumbles_lost <dbl>, site_pts <dbl>, site_fppg <dbl>, src_id <chr>,
#   pass_09_tds <dbl>, pass_1019_tds <dbl>, pass_2029_tds <dbl>,
#   pass_3039_tds <dbl>, pass_4049_tds <dbl>, pass_50_tds <dbl>, ...
```

```
players_projections_seasonal_raw %>%
  bind_rows() %>%
  select(player, pos, data_src, everything()) %>%
  arrange(player, id, pos, data_src) %>%
  group_by(id, pos, data_src) %>%
```

```
  filter(n() > 1, !is.na(id)) %>%
  head()
```

```
# A tibble: 0 x 130
# Groups:   id, pos, data_src [0]
# i 130 variables: player <chr>, pos <chr>, data_src <chr>, team <chr>,
#   games <int>, pass_att <dbl>, pass_comp <dbl>, pass_yds <dbl>,
#   pass_yds_g <dbl>, pass_tds <dbl>, pass_int <dbl>, pass_rate <dbl>,
#   rush_att <dbl>, rush_yds <dbl>, rush_avg <dbl>, rush_tds <dbl>,
#   fumbles_lost <dbl>, site_pts <dbl>, site_fppg <dbl>, src_id <chr>,
#  id <chr>, pass_09_tds <dbl>, pass_1019_tds <dbl>, pass_2029_tds <dbl>,
#   pass_3039_tds <dbl>, pass_4049_tds <dbl>, pass_50_tds <dbl>, ...
```

4.3.26.2 Specify League Scoring Settings

First, create a scoring object using the default scoring object:

```
scoring_obj_default <- ffanalytics::scoring
```

View the default scoring settings:

```
scoring_obj_default
```

```
$pass
$pass$pass_att
[1] 0

$pass$pass_comp
[1] 0

$pass$pass_inc
[1] 0

$pass$pass_yds
[1] 0.04

$pass$pass_tds
[1] 4

$pass$pass_int
[1] -3

$pass$pass_40_yds
```

```
[1] 0

$pass$pass_300_yds
[1] 0

$pass$pass_350_yds
[1] 0

$pass$pass_400_yds
[1] 0


$rush
$rush$all_pos
[1] TRUE

$rush$rush_yds
[1] 0.1

$rush$rush_att
[1] 0

$rush$rush_40_yds
[1] 0

$rush$rush_tds
[1] 6

$rush$rush_100_yds
[1] 0

$rush$rush_150_yds
[1] 0

$rush$rush_200_yds
[1] 0


$rec
$rec$all_pos
[1] TRUE

$rec$rec
[1] 0
```

```
$rec$rec_yds
[1] 0.1

$rec$rec_tds
[1] 6

$rec$rec_40_yds
[1] 0

$rec$rec_100_yds
[1] 0

$rec$rec_150_yds
[1] 0

$rec$rec_200_yds
[1] 0


$misc
$misc$all_pos
[1] TRUE

$misc$fumbles_lost
[1] -3

$misc$fumbles_total
[1] 0

$misc$sacks
[1] 0

$misc$two_pts
[1] 2


$kick
$kick$xp
[1] 1

$kick$fg_0019
[1] 3

$kick$fg_2029
[1] 3
```

```
$kick$fg_3039
[1] 3

$kick$fg_4049
[1] 4

$kick$fg_50
[1] 5

$kick$fg_miss
[1] 0


$ret
$ret$all_pos
[1] TRUE

$ret$return_tds
[1] 6

$ret$return_yds
[1] 0


$idp
$idp$all_pos
[1] TRUE

$idp$idp_solo
[1] 1

$idp$idp_asst
[1] 0.5

$idp$idp_sack
[1] 2

$idp$idp_int
[1] 3

$idp$idp_fum_force
[1] 3

$idp$idp_fum_rec
```

```
[1] 2

$idp$idp_pd
[1] 1

$idp$idp_td
[1] 6

$idp$idp_safety
[1] 2


$dst
$dst$dst_fum_rec
[1] 2

$dst$dst_int
[1] 2

$dst$dst_safety
[1] 2

$dst$dst_sacks
[1] 1

$dst$dst_td
[1] 6

$dst$dst_blk
[1] 1.5

$dst$dst_ret_yds
[1] 0

$dst$dst_pts_allowed
[1] 0


$pts_bracket
$pts_bracket[[1]]
$pts_bracket[[1]]$threshold
[1] 0

$pts_bracket[[1]]$points
[1] 10
```

```
$pts_bracket[[2]]
$pts_bracket[[2]]$threshold
[1] 6

$pts_bracket[[2]]$points
[1] 7


$pts_bracket[[3]]
$pts_bracket[[3]]$threshold
[1] 20

$pts_bracket[[3]]$points
[1] 4


$pts_bracket[[4]]
$pts_bracket[[4]]$threshold
[1] 34

$pts_bracket[[4]]$points
[1] 0


$pts_bracket[[5]]
$pts_bracket[[5]]$threshold
[1] 99

$pts_bracket[[5]]$points
[1] -4
```

Now, modify the scoring settings to match our league settings:

```
scoring_obj <- scoring_obj_default

scoring_obj$pass$pass_int <- -2
scoring_obj$misc$fumbles_lost <- -2
scoring_obj$idp$idp_fum_force <- -2
```

View our scoring settings:

```r
scoring_obj
```

```
$pass
$pass$pass_att
[1] 0

$pass$pass_comp
[1] 0

$pass$pass_inc
[1] 0

$pass$pass_yds
[1] 0.04

$pass$pass_tds
[1] 4

$pass$pass_int
[1] -2

$pass$pass_40_yds
[1] 0

$pass$pass_300_yds
[1] 0

$pass$pass_350_yds
[1] 0

$pass$pass_400_yds
[1] 0


$rush
$rush$all_pos
[1] TRUE

$rush$rush_yds
[1] 0.1

$rush$rush_att
[1] 0
```

```
$rush$rush_40_yds
[1] 0

$rush$rush_tds
[1] 6

$rush$rush_100_yds
[1] 0

$rush$rush_150_yds
[1] 0

$rush$rush_200_yds
[1] 0


$rec
$rec$all_pos
[1] TRUE

$rec$rec
[1] 0

$rec$rec_yds
[1] 0.1

$rec$rec_tds
[1] 6

$rec$rec_40_yds
[1] 0

$rec$rec_100_yds
[1] 0

$rec$rec_150_yds
[1] 0

$rec$rec_200_yds
[1] 0


$misc
$misc$all_pos
[1] TRUE
```

```
$misc$fumbles_lost
[1] -2

$misc$fumbles_total
[1] 0

$misc$sacks
[1] 0

$misc$two_pts
[1] 2


$kick
$kick$xp
[1] 1

$kick$fg_0019
[1] 3

$kick$fg_2029
[1] 3

$kick$fg_3039
[1] 3

$kick$fg_4049
[1] 4

$kick$fg_50
[1] 5

$kick$fg_miss
[1] 0


$ret
$ret$all_pos
[1] TRUE

$ret$return_tds
[1] 6

$ret$return_yds
```

```
[1] 0


$idp
$idp$all_pos
[1] TRUE

$idp$idp_solo
[1] 1

$idp$idp_asst
[1] 0.5

$idp$idp_sack
[1] 2

$idp$idp_int
[1] 3

$idp$idp_fum_force
[1] -2

$idp$idp_fum_rec
[1] 2

$idp$idp_pd
[1] 1

$idp$idp_td
[1] 6

$idp$idp_safety
[1] 2


$dst
$dst$dst_fum_rec
[1] 2

$dst$dst_int
[1] 2

$dst$dst_safety
[1] 2
```

```
$dst$dst_sacks
[1] 1

$dst$dst_td
[1] 6

$dst$dst_blk
[1] 1.5

$dst$dst_ret_yds
[1] 0

$dst$dst_pts_allowed
[1] 0


$pts_bracket
$pts_bracket[[1]]
$pts_bracket[[1]]$threshold
[1] 0

$pts_bracket[[1]]$points
[1] 10


$pts_bracket[[2]]
$pts_bracket[[2]]$threshold
[1] 6

$pts_bracket[[2]]$points
[1] 7


$pts_bracket[[3]]
$pts_bracket[[3]]$threshold
[1] 20

$pts_bracket[[3]]$points
[1] 4


$pts_bracket[[4]]
$pts_bracket[[4]]$threshold
[1] 34
```

```
$pts_bracket[[4]]$points
[1] 0


$pts_bracket[[5]]
$pts_bracket[[5]]$threshold
[1] 99

$pts_bracket[[5]]$points
[1] -4
```

4.3.26.3 Calculate Projected Points

Calculate projected points by source:

```
players_projectedPoints_seasonal <- ffanalytics:::impute_and_score_sources(
  data_result = players_projections_seasonal_raw,
  scoring_rules = scoring_obj)
```

Calculate projected statistics and points, averaged across sources:

```
players_projectedStatsAverage_seasonal <- ffanalytics::projections_table(
  players_projections_seasonal_raw,
  scoring_rules = scoring_obj,
  return_raw_stats = TRUE)

players_projectedPointsAverage_seasonal <- ffanalytics::projections_table(
  players_projections_seasonal_raw,
  scoring_rules = scoring_obj,
  return_raw_stats = FALSE)
```

The `players_projectedPoints_seasonal`, `players_projectedStatsAverage_seasonal`, and `players_projectedPointsAverage_seasonal` objects are in `player-average type-position` form. That is, each row should be uniquely identified by the combination of `id`, `avg_type` and `pos` (or `position`).

Let's check for duplicate `player-position-projection source` instances:

```
players_projectedPoints_seasonal %>%
  bind_rows() %>%
  select(id, pos, data_src, everything()) %>%
  arrange(player, id, pos, data_src) %>%
  group_by(id, pos, data_src) %>%
  filter(n() > 1, !is.na(id)) %>%
  head()
```

```
# A tibble: 0 x 137
# Groups:   id, pos, data_src [0]
# i 137 variables: id <chr>, pos <chr>, data_src <chr>, player <chr>,
#  team <chr>, games <int>, pass_att <dbl>, pass_comp <dbl>, pass_yds <dbl>,
#   pass_yds_g <dbl>, pass_tds <dbl>, pass_int <dbl>, pass_rate <dbl>,
#   rush_att <dbl>, rush_yds <dbl>, rush_avg <dbl>, rush_tds <dbl>,
#   fumbles_lost <dbl>, site_pts <dbl>, site_fppg <dbl>, src_id <chr>,
#   pass_09_tds <dbl>, pass_1019_tds <dbl>, pass_2029_tds <dbl>,
#   pass_3039_tds <dbl>, pass_4049_tds <dbl>, pass_50_tds <dbl>, ...
```

```
players_projectedStatsAverage_seasonal %>%
  group_by(id, avg_type, position) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 59
# Groups:   id, avg_type, position [0]
# i 59 variables: position <chr>, id <chr>, avg_type <chr>, pass_yds <dbl>,
#  pass_yds_sd <dbl>, pass_tds <dbl>, pass_tds_sd <dbl>, pass_int <dbl>,
#  pass_int_sd <dbl>, rush_yds <dbl>, rush_yds_sd <dbl>, rush_tds <dbl>,
#   rush_tds_sd <dbl>, fumbles_lost <dbl>, fumbles_lost_sd <dbl>,
#   rec_yds <dbl>, rec_yds_sd <dbl>, rec_tds <dbl>, rec_tds_sd <dbl>,
#  return_tds <dbl>, return_tds_sd <dbl>, fg_0019 <dbl>, fg_0019_sd <dbl>,
#  fg_2029 <dbl>, fg_2029_sd <dbl>, fg_3039 <dbl>, fg_3039_sd <dbl>, ...
```

```
players_projectedPointsAverage_seasonal %>%
  group_by(id, avg_type, pos) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 16
# Groups:   id, avg_type, pos [0]
# i 16 variables: avg_type <chr>, id <chr>, pos <chr>, points <dbl>,
#  sd_pts <dbl>, dropoff <dbl>, floor <dbl>, ceiling <dbl>, points_vor <dbl>,
#   floor_vor <dbl>, ceiling_vor <dbl>, rank <int>, floor_rank <int>,
#   ceiling_rank <int>, pos_rank <int>, tier <int>
```

Let's merge the two averaged projected statistics and points objects:

```
players_projections_seasonal_average <- full_join(
  players_projectedPointsAverage_seasonal,
  players_projectedStatsAverage_seasonal,
  by = c("id","avg_type","pos" = "position")
)
```

Let's again check for duplicate `player-position-projection source` instances:

```
players_projections_seasonal_average %>%
  group_by(id, avg_type, pos) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 72
# Groups:   id, avg_type, pos [0]
# i 72 variables: avg_type <chr>, id <chr>, pos <chr>, points <dbl>,
#  sd_pts <dbl>, dropoff <dbl>, floor <dbl>, ceiling <dbl>, points_vor <dbl>,
#   floor_vor <dbl>, ceiling_vor <dbl>, rank <int>, floor_rank <int>,
#   ceiling_rank <int>, pos_rank <int>, tier <int>, pass_yds <dbl>,
#  pass_yds_sd <dbl>, pass_tds <dbl>, pass_tds_sd <dbl>, pass_int <dbl>,
#  pass_int_sd <dbl>, rush_yds <dbl>, rush_yds_sd <dbl>, rush_tds <dbl>,
#   rush_tds_sd <dbl>, fumbles_lost <dbl>, fumbles_lost_sd <dbl>, ...
```

4.3.26.4 Add Additional Player Information

```
players_projections_seasonal_average <- players_projections_seasonal_average %>%
  add_ecr() %>%
  add_adp() %>%
  add_aav() %>%
  add_uncertainty() %>%
  add_player_info()
```

```
save(
  players_projectedPoints_seasonal,
  file = "./data/players_projectedPoints_seasonal.RData"
)

save(
  players_projections_seasonal_average,
  file = "./data/players_projections_seasonal_average.RData"
)
```

The data file is saved in the project repository and can be loaded using the following command:

```
load(file = "./data/players_projectedPoints_seasonal.RData")
load(file = "./data/players_projections_seasonal_average.RData")
```

4.4 Calculations

4.4.1 Historical Actual Player Statistics

In addition to week-by-week actual player statistics, we can also compute historical actual player statistics as a function of different timeframes, including season-by-season and career statistics.

4.4.1.1 Career Statistics

First, we can compute the players' career statistics using the `calculate_player_stats()`, `calculate_player_stats_def()`, and `calculate_player_stats_kicking()` functions from the `nflfastR` package for offensive players, defensive players, and kickers, respectively.

> **Note 5: Calculating players' career statistics**
>
> Note: the following code takes a while to run.

```
nfl_actualStats_offense_career_raw <- nflfastR::calculate_player_stats(
  nfl_pbp,
  weekly = FALSE)

nfl_actualStats_defense_career_raw <- nflfastR::calculate_player_stats_def(
  nfl_pbp,
  weekly = FALSE)

nfl_actualStats_kicking_career_raw <- nflfastR::calculate_player_stats_kicking(
  nfl_pbp,
  weekly = FALSE)
```

```
save(
  nfl_actualStats_offense_career_raw, nfl_actualStats_defense_career_raw, nfl_actualStats_kicking_
  file = "./data/nfl_actualStats_career_raw.RData"
)
```

The `nfl_actualStats_career` objects are in `player` form. That is, each row should be uniquely identified by the combination of `player_id`. Let's rearrange the data accordingly:

```
nfl_actualStats_offense_career <- nfl_actualStats_offense_career_raw %>%
  arrange(player_display_name, player_id)

nfl_actualStats_defense_career <- nfl_actualStats_defense_career_raw %>%
  arrange(player_display_name, player_id)

nfl_actualStats_kicking_career <- nfl_actualStats_kicking_career_raw %>%
  arrange(player_display_name, player_id)
```

Let's check for duplicate `player` instances:

```
nfl_actualStats_offense_career %>%
  group_by(player_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 50
# Groups:   player_id [0]
# i 50 variables: player_id <chr>, player_name <chr>,
#   player_display_name <chr>, position <chr>, position_group <chr>,
#  headshot_url <chr>, games <int>, recent_team <chr>, completions <int>,
#   attempts <int>, passing_yards <dbl>, passing_tds <int>,
#  interceptions <dbl>, sacks <dbl>, sack_yards <dbl>, sack_fumbles <int>,
#   sack_fumbles_lost <int>, passing_air_yards <dbl>,
#   passing_yards_after_catch <dbl>, passing_first_downs <dbl>, ...
```

```
nfl_actualStats_defense_career %>%
  group_by(player_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 6 x 30
# Groups:   player_id [2]
  player_id  player_name player_display_name games position position_group
  <chr>      <chr>       <chr>               <int> <chr>    <chr>
1 00-0032889 A.Robinson  A'Shawn Robinson       57 DT       DL
2 00-0032889 A.Robinson  A'Shawn Robinson       40 DT       DL
3 00-0032889 A.Robinson  A'Shawn Robinson       16 DT       DL
4 00-0030228 A.Bouye     A.J. Bouye             10 CB       DB
5 00-0030228 A.Bouye     A.J. Bouye              7 CB       DB
6 00-0030228 A.Bouye     A.J. Bouye             42 CB       DB
# i 24 more variables: headshot_url <chr>, team <chr>, def_tackles <int>,
#   def_tackles_solo <int>, def_tackles_with_assist <int>,
```

```
#   def_tackle_assists <int>, def_tackles_for_loss <int>,
#   def_tackles_for_loss_yards <dbl>, def_fumbles_forced <int>,
#   def_sacks <dbl>, def_sack_yards <dbl>, def_qb_hits <dbl>,
#   def_interceptions <dbl>, def_interception_yards <dbl>,
#   def_pass_defended <dbl>, def_tds <dbl>, def_fumbles <dbl>, ...
```

```r
nfl_actualStats_kicking_career %>%
  group_by(player_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 6 x 42
# Groups:   player_id [3]
 player_id  team player_name player_display_name games position position_group
  <chr>      <chr> <chr>       <chr>               <int> <chr>    <chr>
1 00-0021080 BAL   <NA>        Aaron Elling            1 K        SPEC
2 00-0021080 MIN   <NA>        Aaron Elling           16 K        SPEC
3 00-0021080 TEN   <NA>        Aaron Elling            1 K        SPEC
4 00-0016919 IND   A.Vinatieri Adam Vinatieri        219 K        SPEC
5 00-0016919 NE    A.Vinatieri Adam Vinatieri        122 K        SPEC
6 00-0032870 DET   A.Rosas     Aldrick Rosas           1 K        SPEC
# i 35 more variables: headshot_url <chr>, fg_made <int>, fg_att <dbl>,
#   fg_missed <int>, fg_blocked <int>, fg_long <dbl>, fg_pct <dbl>,
#   fg_made_0_19 <int>, fg_made_20_29 <int>, fg_made_30_39 <int>,
#   fg_made_40_49 <int>, fg_made_50_59 <int>, fg_made_60_ <int>,
#   fg_missed_0_19 <int>, fg_missed_20_29 <int>, fg_missed_30_39 <int>,
#   fg_missed_40_49 <int>, fg_missed_50_59 <int>, fg_missed_60_ <int>,
#   fg_made_list <chr>, fg_missed_list <chr>, fg_blocked_list <chr>, ...
```

Let's do some data cleanup:

```r
# Sum statistics across a player's years with multiple teams
nfl_actualStats_defense_career <- nfl_actualStats_defense_career %>%
  group_by(player_id) %>%
  summarise(
    across(where(is.numeric), ~ sum(.x, na.rm = TRUE)),
    .groups = "drop"
  )

nfl_actualStats_kicking_career <- nfl_actualStats_kicking_career %>%
  group_by(player_id) %>%
  summarise(
    across(where(is.numeric), ~ sum(.x, na.rm = TRUE)),
    .groups = "drop"
```

```
  )

# Re-calculate percentage stats
nfl_actualStats_kicking_career <- nfl_actualStats_kicking_career %>%
  mutate(
    fg_pct = fg_made / fg_att,
    pat_pct = pat_made / pat_att
  )

# Merge data back with player info
nfl_actualStats_defense_career_raw_playerInfo <- nfl_actualStats_defense_career_raw %>%
  select(player_id, player_name, player_display_name, position, position_group, team, headshot_url
  unique() %>%
  group_by(player_id) %>%
  slice_head(n = 1) %>%
  ungroup()

nfl_actualStats_kicking_career_raw_playerInfo <- nfl_actualStats_kicking_career_raw %>%
  select(player_id, player_name, player_display_name, position, position_group, team, headshot_url
  unique() %>%
  group_by(player_id) %>%
  slice_head(n = 1) %>%
  ungroup()

nfl_actualStats_defense_career <- nfl_actualStats_defense_career_raw_playerInfo %>%
  right_join(
    nfl_actualStats_defense_career,
    by = "player_id"
  )

nfl_actualStats_kicking_career <- nfl_actualStats_kicking_career_raw_playerInfo %>%
  right_join(
    nfl_actualStats_kicking_career,
    by = "player_id"
  )

# Rearrange data
nfl_actualStats_defense_career <- nfl_actualStats_defense_career %>%
  arrange(player_display_name, player_id)

nfl_actualStats_kicking_career <- nfl_actualStats_kicking_career %>%
  arrange(player_display_name, player_id)
```

Let's check again for duplicate `player` instances:

```r
nfl_actualStats_offense_career %>%
  group_by(player_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 50
# Groups:   player_id [0]
# i 50 variables: player_id <chr>, player_name <chr>,
#   player_display_name <chr>, position <chr>, position_group <chr>,
#  headshot_url <chr>, games <int>, recent_team <chr>, completions <int>,
#   attempts <int>, passing_yards <dbl>, passing_tds <int>,
#  interceptions <dbl>, sacks <dbl>, sack_yards <dbl>, sack_fumbles <int>,
#   sack_fumbles_lost <int>, passing_air_yards <dbl>,
#   passing_yards_after_catch <dbl>, passing_first_downs <dbl>, ...
```

```r
nfl_actualStats_defense_career %>%
  group_by(player_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 30
# Groups:   player_id [0]
# i 30 variables: player_id <chr>, player_name <chr>,
#   player_display_name <chr>, position <chr>, position_group <chr>,
#   team <chr>, headshot_url <chr>, games <int>, def_tackles <int>,
#   def_tackles_solo <int>, def_tackles_with_assist <int>,
#   def_tackle_assists <int>, def_tackles_for_loss <int>,
#   def_tackles_for_loss_yards <dbl>, def_fumbles_forced <int>,
#   def_sacks <dbl>, def_sack_yards <dbl>, def_qb_hits <dbl>, ...
```

```r
nfl_actualStats_kicking_career %>%
  group_by(player_id) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 38
# Groups:   player_id [0]
# i 38 variables: player_id <chr>, player_name <chr>,
#   player_display_name <chr>, position <chr>, position_group <chr>,
#  team <chr>, headshot_url <chr>, games <int>, fg_made <int>, fg_att <dbl>,
#   fg_missed <int>, fg_blocked <int>, fg_long <dbl>, fg_pct <dbl>,
#   fg_made_0_19 <int>, fg_made_20_29 <int>, fg_made_30_39 <int>,
#   fg_made_40_49 <int>, fg_made_50_59 <int>, fg_made_60_ <int>,
#  fg_missed_0_19 <int>, fg_missed_20_29 <int>, fg_missed_30_39 <int>, ...
```

4.4.1.2 Season-by-Season Statistics

Second, we can compute the players' season-by-season statistics.

```
seasons <- unique(nfl_pbp$season)

nfl_pbp_seasonalList <- list()
nfl_actualStats_offense_seasonalList <- list()
nfl_actualStats_defense_seasonalList <- list()
nfl_actualStats_kicking_seasonalList <- list()
```

> **Note 6: Calculating players' season-by-season statistics**
>
> Note: the following code takes a while to run.

```
pb <- txtProgressBar(
  min = 0,
  max = length(seasons),
  style = 3)

for(i in 1:length(seasons)){
  # Subset play-by-play data by season
  nfl_pbp_seasonalList[[i]] <- nfl_pbp %>%
    dplyr::filter(season == seasons[i])

  # Compute actual statistics by season
  nfl_actualStats_offense_seasonalList[[i]] <-
    nflfastR::calculate_player_stats(
      nfl_pbp_seasonalList[[i]],
      weekly = FALSE)

  nfl_actualStats_defense_seasonalList[[i]] <-
    nflfastR::calculate_player_stats_def(
      nfl_pbp_seasonalList[[i]],
      weekly = FALSE)

  nfl_actualStats_kicking_seasonalList[[i]] <-
    nflfastR::calculate_player_stats_kicking(
      nfl_pbp_seasonalList[[i]],
      weekly = FALSE)

  nfl_actualStats_offense_seasonalList[[i]]$season <- seasons[i]
  nfl_actualStats_defense_seasonalList[[i]]$season <- seasons[i]
  nfl_actualStats_kicking_seasonalList[[i]]$season <- seasons[i]
```

```
  print(
    paste("Completed computing projections for season: ", seasons[i], sep = ""))

  # Update the progress bar
  setTxtProgressBar(pb, i)
}

# Close the progress bar
close(pb)

nfl_actualStats_offense_seasonal_raw <- nfl_actualStats_offense_seasonalList %>%
  dplyr::bind_rows()
nfl_actualStats_defense_seasonal_raw <- nfl_actualStats_defense_seasonalList %>%
  dplyr::bind_rows()
nfl_actualStats_kicking_seasonal_raw <- nfl_actualStats_kicking_seasonalList %>%
  dplyr::bind_rows()
```

```
save(
  nfl_actualStats_offense_seasonal_raw, nfl_actualStats_defense_seasonal_raw, nfl_actualStats_kick
  file = "./data/nfl_actualStats_seasonal_raw.RData"
)
```

The `nfl_actualStats_seasonal` objects are in `player-season` form. That is, each row should be uniquely identified by the combination of `player_id` and `season`. Let's rearrange the data accordingly:

```
nfl_actualStats_offense_seasonal <- nfl_actualStats_offense_seasonal_raw %>%
  rename(team = recent_team) %>%
  select(player_id, season, everything()) %>%
  arrange(player_display_name, player_id, season)

nfl_actualStats_defense_seasonal <- nfl_actualStats_defense_seasonal_raw %>%
  select(player_id, season, everything()) %>%
  arrange(player_display_name, player_id, season)

nfl_actualStats_kicking_seasonal <- nfl_actualStats_kicking_seasonal_raw %>%
  select(player_id, season, everything()) %>%
  arrange(player_display_name, player_id, season)
```

Let's check for duplicate `player` instances:

```
nfl_actualStats_offense_seasonal %>%
  group_by(player_id, season) %>%
```

```
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 51
# Groups:   player_id, season [0]
# i 51 variables: player_id <chr>, season <int>, player_name <chr>,
#   player_display_name <chr>, position <chr>, position_group <chr>,
#   headshot_url <chr>, games <int>, team <chr>, completions <int>,
#   attempts <int>, passing_yards <dbl>, passing_tds <int>,
#  interceptions <dbl>, sacks <dbl>, sack_yards <dbl>, sack_fumbles <int>,
#   sack_fumbles_lost <int>, passing_air_yards <dbl>,
#   passing_yards_after_catch <dbl>, passing_first_downs <dbl>, ...
```

```
nfl_actualStats_defense_seasonal %>%
  group_by(player_id, season) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 6 x 31
# Groups:   player_id, season [3]
  player_id season player_name player_display_name games position position_group
  <chr>      <int> <chr>       <chr>               <int> <chr>    <chr>
1 00-00356~   2021 A.Brown     A.J. Brown              1 WR       WR
2 00-00356~   2021 A.Brown     A.J. Brown              1 WR       WR
3 00-00356~   2021 A.Brown     A.J. Brown              1 WR       WR
4 00-00356~   2022 A.Brown     A.J. Brown              1 WR       WR
5 00-00356~   2022 A.Brown     A.J. Brown              1 WR       WR
6 00-00322~   2016 A.Cann      A.J. Cann               1 G        OL
# i 24 more variables: headshot_url <chr>, team <chr>, def_tackles <int>,
#   def_tackles_solo <int>, def_tackles_with_assist <int>,
#   def_tackle_assists <int>, def_tackles_for_loss <int>,
#   def_tackles_for_loss_yards <dbl>, def_fumbles_forced <int>,
#   def_sacks <dbl>, def_sack_yards <dbl>, def_qb_hits <dbl>,
#   def_interceptions <dbl>, def_interception_yards <dbl>,
#   def_pass_defended <dbl>, def_tds <dbl>, def_fumbles <dbl>, ...
```

```
nfl_actualStats_kicking_seasonal %>%
  group_by(player_id, season) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 6 x 43
# Groups:   player_id, season [3]
```

```
  player_id  season team  player_name player_display_name games position
  <chr>       <int> <chr> <chr>       <chr>               <int> <chr>
1 00-0032870   2021 DET   A.Rosas     Aldrick Rosas           1 K
2 00-0032870   2021 NO    A.Rosas     Aldrick Rosas           4 K
3 00-0035145   2020 CIN   A.Seibert   Austin Seibert          4 K
4 00-0035145   2020 CLE   A.Seibert   Austin Seibert          1 K
5 00-0020972   2009 BAL   <NA>        Billy Cundiff           9 K
6 00-0020972   2009 CLE   <NA>        Billy Cundiff           5 K
# i 36 more variables: position_group <chr>, headshot_url <chr>, fg_made <int>,
#   fg_att <dbl>, fg_missed <int>, fg_blocked <int>, fg_long <dbl>,
#  fg_pct <dbl>, fg_made_0_19 <int>, fg_made_20_29 <int>, fg_made_30_39 <int>,
#   fg_made_40_49 <int>, fg_made_50_59 <int>, fg_made_60_ <int>,
#   fg_missed_0_19 <int>, fg_missed_20_29 <int>, fg_missed_30_39 <int>,
#   fg_missed_40_49 <int>, fg_missed_50_59 <int>, fg_missed_60_ <int>,
#  fg_made_list <chr>, fg_missed_list <chr>, fg_blocked_list <chr>, ...
```

Let's do some data cleanup:

```r
# Sum statistics across a player's years with multiple teams
nfl_actualStats_defense_seasonal <- nfl_actualStats_defense_seasonal %>%
  group_by(player_id, season) %>%
  summarise(
    across(where(is.numeric), ~ sum(.x, na.rm = TRUE)),
    .groups = "drop"
  )

nfl_actualStats_kicking_seasonal <- nfl_actualStats_kicking_seasonal %>%
  group_by(player_id, season) %>%
  summarise(
    across(where(is.numeric), ~ sum(.x, na.rm = TRUE)),
    .groups = "drop"
  )

# Re-calculate percentage stats
nfl_actualStats_kicking_seasonal <- nfl_actualStats_kicking_seasonal %>%
  mutate(
    fg_pct = fg_made / fg_att,
    pat_pct = pat_made / pat_att
  )

# Merge data back with player info
nfl_actualStats_defense_seasonal_raw_playerInfo <- nfl_actualStats_defense_seasonal_raw %>%
  select(player_id, season, player_name, player_display_name, position, position_group, team, head
  unique() %>%
  group_by(player_id) %>%
```

```
  slice_head(n = 1) %>%
  ungroup()

nfl_actualStats_kicking_seasonal_raw_playerInfo <- nfl_actualStats_kicking_seasonal_raw %>%
  select(player_id, season, player_name, player_display_name, position, position_group, team, head
  unique() %>%
  group_by(player_id) %>%
  slice_head(n = 1) %>%
  ungroup()

nfl_actualStats_defense_seasonal <- nfl_actualStats_defense_seasonal_raw_playerInfo %>%
  right_join(
    nfl_actualStats_defense_seasonal,
    by = c("player_id","season")
  )

nfl_actualStats_kicking_seasonal <- nfl_actualStats_kicking_seasonal_raw_playerInfo %>%
  right_join(
    nfl_actualStats_kicking_seasonal,
    by = c("player_id","season")
  )

# Rearrange data
nfl_actualStats_defense_seasonal <- nfl_actualStats_defense_seasonal %>%
  select(player_id, season, everything()) %>%
  arrange(player_display_name, player_id, season)

nfl_actualStats_kicking_seasonal <- nfl_actualStats_kicking_seasonal %>%
  select(player_id, season, everything()) %>%
  arrange(player_display_name, player_id, season)
```

Let's check again for duplicate `player` instances:

```
nfl_actualStats_offense_seasonal %>%
  group_by(player_id, season) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 51
# Groups:   player_id, season [0]
# i 51 variables: player_id <chr>, season <int>, player_name <chr>,
#   player_display_name <chr>, position <chr>, position_group <chr>,
#   headshot_url <chr>, games <int>, team <chr>, completions <int>,
#   attempts <int>, passing_yards <dbl>, passing_tds <int>,
```

```
#  interceptions <dbl>, sacks <dbl>, sack_yards <dbl>, sack_fumbles <int>,
#   sack_fumbles_lost <int>, passing_air_yards <dbl>,
#   passing_yards_after_catch <dbl>, passing_first_downs <dbl>, ...
```

```
nfl_actualStats_defense_seasonal %>%
  group_by(player_id, season) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 31
# Groups:   player_id, season [0]
# i 31 variables: player_id <chr>, season <int>, player_name <chr>,
#   player_display_name <chr>, position <chr>, position_group <chr>,
#   team <chr>, headshot_url <chr>, games <int>, def_tackles <int>,
#   def_tackles_solo <int>, def_tackles_with_assist <int>,
#   def_tackle_assists <int>, def_tackles_for_loss <int>,
#   def_tackles_for_loss_yards <dbl>, def_fumbles_forced <int>,
#   def_sacks <dbl>, def_sack_yards <dbl>, def_qb_hits <dbl>, ...
```

```
nfl_actualStats_kicking_seasonal %>%
  group_by(player_id, season) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 39
# Groups:   player_id, season [0]
# i 39 variables: player_id <chr>, season <int>, player_name <chr>,
#   player_display_name <chr>, position <chr>, position_group <chr>,
#  team <chr>, headshot_url <chr>, games <int>, fg_made <int>, fg_att <dbl>,
#   fg_missed <int>, fg_blocked <int>, fg_long <dbl>, fg_pct <dbl>,
#   fg_made_0_19 <int>, fg_made_20_29 <int>, fg_made_30_39 <int>,
#   fg_made_40_49 <int>, fg_made_50_59 <int>, fg_made_60_ <int>,
#  fg_missed_0_19 <int>, fg_missed_20_29 <int>, fg_missed_30_39 <int>, ...
```

4.4.1.3 Week-by-Week Statistics

We already load players' week-by-week statistics above. Nevertheless, we could compute players' weekly statistics from the play-by-play data using the following syntax:

```
nfl_actualStats_offense_weekly <- nflfastR::calculate_player_stats(
  nfl_pbp,
  weekly = TRUE)
```

```
nfl_actualStats_defense_weekly <- nflfastR::calculate_player_stats_def(
  nfl_pbp,
  weekly = TRUE)

nfl_actualStats_kicking_weekly <- nflfastR::calculate_player_stats_kicking(
  nfl_pbp,
  weekly = TRUE)
```

4.4.2 Historical Actual Fantasy Points

Specify scoring settings:

4.4.2.1 Weekly

4.4.2.2 Seasonal

4.4.2.3 Career

4.4.3 Player Age and Experience

4.4.3.1 Weekly

```
# Reshape from wide to long format
nfl_actualStats_offense_weekly_long <- nfl_actualStats_offense_weekly %>%
  tidyr::pivot_longer(
    cols = c(team, opponent_team),
    names_to = "role",
    values_to = "team")

# Perform separate inner join operations for the home_team and away_team
nfl_actualStats_offense_weekly_home <- dplyr::inner_join(
  nfl_actualStats_offense_weekly_long,
  nfl_schedules,
  by = c("season","week","team" = "home_team")) %>%
  mutate(home_away = "home_team")

nfl_actualStats_offense_weekly_away <- dplyr::inner_join(
  nfl_actualStats_offense_weekly_long,
  nfl_schedules,
  by = c("season","week","team" = "away_team")) %>%
  mutate(home_away = "away_team")
```

```
nfl_actualStats_defense_weekly_home <- dplyr::inner_join(
  nfl_actualStats_defense_weekly,
  nfl_schedules,
  by = c("season","week","team" = "home_team")) %>%
  mutate(home_away = "home_team")

nfl_actualStats_defense_weekly_away <- dplyr::inner_join(
  nfl_actualStats_defense_weekly,
  nfl_schedules,
  by = c("season","week","team" = "away_team")) %>%
  mutate(home_away = "away_team")

nfl_actualStats_kicking_weekly_home <- dplyr::inner_join(
  nfl_actualStats_kicking_weekly,
  nfl_schedules,
  by = c("season","week","team" = "home_team")) %>%
  mutate(home_away = "home_team")

nfl_actualStats_kicking_weekly_away <- dplyr::inner_join(
  nfl_actualStats_kicking_weekly,
  nfl_schedules,
  by = c("season","week","team" = "away_team")) %>%
  mutate(home_away = "away_team")

# Combine the results of the join operations
nfl_actualStats_offense_weekly_schedules_long <- dplyr::bind_rows(
  nfl_actualStats_offense_weekly_home,
  nfl_actualStats_offense_weekly_away)

nfl_actualStats_defense_weekly_schedules_long <- dplyr::bind_rows(
  nfl_actualStats_defense_weekly_home,
  nfl_actualStats_defense_weekly_away)

nfl_actualStats_kicking_weekly_schedules_long <- dplyr::bind_rows(
  nfl_actualStats_kicking_weekly_home,
  nfl_actualStats_kicking_weekly_away)

# Reshape from long to wide
player_game_gameday_offense <- nfl_actualStats_offense_weekly_schedules_long %>%
  dplyr::distinct(player_id, season, week, game_id, home_away, team, gameday) %>% #, .keep_all = T
  tidyr::pivot_wider(
    names_from = home_away,
    values_from = team)
```

```
player_game_gameday_defense <- nfl_actualStats_defense_weekly_schedules_long %>%
  dplyr::distinct(player_id, season, week, game_id, home_away, team, gameday) %>% #, .keep_all = T
  tidyr::pivot_wider(
    names_from = home_away,
    values_from = team)

player_game_gameday_kicking <- nfl_actualStats_kicking_weekly_schedules_long %>%
  dplyr::distinct(player_id, season, week, game_id, home_away, team, gameday) %>% #, .keep_all = T
  tidyr::pivot_wider(
    names_from = home_away,
    values_from = team)

# Merge player birthdate and the game date
player_game_birthdate_gameday_offense <- dplyr::left_join(
  player_game_gameday_offense,
  unique(nfl_players[,c("gsis_id","birth_date")]),
  by = c("player_id" = "gsis_id")
)

player_game_birthdate_gameday_defense <- dplyr::left_join(
  player_game_gameday_defense,
  unique(nfl_players[,c("gsis_id","birth_date")]),
  by = c("player_id" = "gsis_id")
)

player_game_birthdate_gameday_kicking <- dplyr::left_join(
  player_game_gameday_kicking,
  unique(nfl_players[,c("gsis_id","birth_date")]),
  by = c("player_id" = "gsis_id")
)

player_game_birthdate_gameday_offense$birth_date <- lubridate::ymd(player_game_birthdate_gameday_o
player_game_birthdate_gameday_offense$gameday <- lubridate::ymd(player_game_birthdate_gameday_offe

player_game_birthdate_gameday_defense$birth_date <- lubridate::ymd(player_game_birthdate_gameday_d
player_game_birthdate_gameday_defense$gameday <- lubridate::ymd(player_game_birthdate_gameday_defe

player_game_birthdate_gameday_kicking$birth_date <- lubridate::ymd(player_game_birthdate_gameday_k
player_game_birthdate_gameday_kicking$gameday <- lubridate::ymd(player_game_birthdate_gameday_kick

# Calculate player's age for a given week as the difference between their birthdate and the game d
player_game_birthdate_gameday_offense$age <- lubridate::interval(
  start = player_game_birthdate_gameday_offense$birth_date,
  end = player_game_birthdate_gameday_offense$gameday
```

```
) %>%
  lubridate::time_length(unit = "years")

player_game_birthdate_gameday_defense$age <- lubridate::interval(
  start = player_game_birthdate_gameday_defense$birth_date,
  end = player_game_birthdate_gameday_defense$gameday
) %>%
  lubridate::time_length(unit = "years")

player_game_birthdate_gameday_kicking$age <- lubridate::interval(
  start = player_game_birthdate_gameday_kicking$birth_date,
  end = player_game_birthdate_gameday_kicking$gameday
) %>%
  lubridate::time_length(unit = "years")

# Merge with Pro Football Reference Data on Player Age by Season
player_game_birthdate_gameday_offense <- player_game_birthdate_gameday_offense %>%
  dplyr::left_join(
    nfl_advancedStatsPFR_seasonal %>% filter(!is.na(gsis_id), !is.na(season), !is.na(age)) %>% sel
    by = c("player_id" = "gsis_id", "season")
  )

player_game_birthdate_gameday_defense <- player_game_birthdate_gameday_defense %>%
  dplyr::left_join(
    nfl_advancedStatsPFR_seasonal %>% filter(!is.na(gsis_id), !is.na(season), !is.na(age)) %>% sel
    by = c("player_id" = "gsis_id", "season")
  )

player_game_birthdate_gameday_kicking <- player_game_birthdate_gameday_kicking %>%
  dplyr::left_join(
    nfl_advancedStatsPFR_seasonal %>% filter(!is.na(gsis_id), !is.na(season), !is.na(age)) %>% sel
    by = c("player_id" = "gsis_id", "season")
  )

# Set age as first non-missing value from calculation above or from PFR
player_game_birthdate_gameday_offense <- player_game_birthdate_gameday_offense %>%
  mutate(age = coalesce(age.x, age.y)) %>%
  select(-age.x, -age.y)

player_game_birthdate_gameday_defense <- player_game_birthdate_gameday_defense %>%
  mutate(age = coalesce(age.x, age.y)) %>%
  select(-age.x, -age.y)

player_game_birthdate_gameday_kicking <- player_game_birthdate_gameday_kicking %>%
```

```
  mutate(age = coalesce(age.x, age.y)) %>%
  select(-age.x, -age.y)

# Calculate ageCentered and ageCenteredQuadratic
player_game_birthdate_gameday_offense$ageCentered20 <- player_game_birthdate_gameday_offense$age -
player_game_birthdate_gameday_offense$ageCentered20Quadratic <- player_game_birthdate_gameday_offe

player_game_birthdate_gameday_defense$ageCentered20 <- player_game_birthdate_gameday_defense$age -
player_game_birthdate_gameday_defense$ageCentered20Quadratic <- player_game_birthdate_gameday_defe

player_game_birthdate_gameday_kicking$ageCentered20 <- player_game_birthdate_gameday_kicking$age -
player_game_birthdate_gameday_kicking$ageCentered20Quadratic <- player_game_birthdate_gameday_kick

# Merge with player info
player_age_offense <- dplyr::left_join(
  player_game_birthdate_gameday_offense,
  nfl_players %>% select(-birth_date, -season, -team_abbr, - team_seq),
  by = c("player_id" = "gsis_id"))

player_age_defense <- dplyr::left_join(
  player_game_birthdate_gameday_defense,
  nfl_players %>% select(-birth_date, -season, -team_abbr, - team_seq),
  by = c("player_id" = "gsis_id"))

player_age_kicking <- dplyr::left_join(
  player_game_birthdate_gameday_kicking,
  nfl_players %>% select(-birth_date, -season, -team_abbr, - team_seq),
  by = c("player_id" = "gsis_id"))

# Add game_id to weekly stats to facilitate merging
nfl_actualStats_game_offense_weekly <- nfl_actualStats_offense_weekly %>%
  dplyr::left_join(
    player_age_offense[,c("season","week","player_id","game_id")],
    by = c("season","week","player_id"))

nfl_actualStats_game_defense_weekly <- nfl_actualStats_defense_weekly %>%
  dplyr::left_join(
    player_age_offense[,c("season","week","player_id","game_id")],
    by = c("season","week","player_id"))

nfl_actualStats_game_kicking_weekly <- nfl_actualStats_kicking_weekly %>%
  dplyr::left_join(
    player_age_offense[,c("season","week","player_id","game_id")],
    by = c("season","week","player_id"))
```

```
# Merge with player weekly stats
player_stats_weekly_offense <- dplyr::full_join(
  player_age_offense %>% select(-position, -position_group),
  nfl_actualStats_game_offense_weekly,
  by = c("season","week","player_id","game_id"))

player_stats_weekly_defense <- dplyr::full_join(
  player_age_defense %>% select(-position, -position_group),
  nfl_actualStats_game_defense_weekly,
  by = c("season","week","player_id","game_id"))

player_stats_weekly_kicking <- dplyr::full_join(
  player_age_kicking %>% select(-position, -position_group),
  nfl_actualStats_game_kicking_weekly,
  by = c("season","week","player_id","game_id"))

player_stats_weekly_offense$total_years_of_experience <- as.integer(player_stats_weekly_offense$ye
player_stats_weekly_defense$total_years_of_experience <- as.integer(player_stats_weekly_defense$ye
player_stats_weekly_kicking$total_years_of_experience <- as.integer(player_stats_weekly_kicking$ye

player_stats_weekly_offense$years_of_experience <- NULL
player_stats_weekly_defense$years_of_experience <- NULL
player_stats_weekly_kicking$years_of_experience <- NULL

distinct_seasons_offense <- player_stats_weekly_offense %>%
  dplyr::select(player_id, season) %>%
  dplyr::distinct() %>%
  dplyr::left_join(
    nfl_players[,c("gsis_id","years_of_experience")],
    by = c("player_id" = "gsis_id")
  ) %>%
  dplyr::mutate(total_years_of_experience = as.integer(years_of_experience)) %>%
  dplyr::select(-years_of_experience)

distinct_seasons_defense <- player_stats_weekly_defense %>%
  dplyr::select(player_id, season) %>%
  dplyr::distinct() %>%
  dplyr::left_join(
    nfl_players[,c("gsis_id","years_of_experience")],
    by = c("player_id" = "gsis_id")
  ) %>%
  dplyr::mutate(total_years_of_experience = as.integer(years_of_experience)) %>%
  dplyr::select(-years_of_experience)
```

```
distinct_seasons_kicking <- player_stats_weekly_kicking %>%
  dplyr::select(player_id, season) %>%
  dplyr::distinct() %>%
  dplyr::left_join(
    nfl_players[,c("gsis_id","years_of_experience")],
    by = c("player_id" = "gsis_id")
  ) %>%
  dplyr::mutate(total_years_of_experience = as.integer(years_of_experience)) %>%
  dplyr::select(-years_of_experience)

years_of_experience_offense <- distinct_seasons_offense %>%
  dplyr::arrange(player_id, -season) %>%
  dplyr::group_by(player_id) %>%
  dplyr::mutate(years_of_experience = first(total_years_of_experience) - (row_number() - 1)) %>%
  dplyr::ungroup()

years_of_experience_defense <- distinct_seasons_defense %>%
  dplyr::arrange(player_id, -season) %>%
  dplyr::group_by(player_id) %>%
  dplyr::mutate(years_of_experience = first(total_years_of_experience) - (row_number() - 1)) %>%
  dplyr::ungroup()

years_of_experience_kicking <- distinct_seasons_kicking %>%
  dplyr::arrange(player_id, -season) %>%
  dplyr::group_by(player_id) %>%
  dplyr::mutate(years_of_experience = first(total_years_of_experience) - (row_number() - 1)) %>%
  dplyr::ungroup()

years_of_experience_offense$years_of_experience[which(years_of_experience_offense$years_of_experie
years_of_experience_defense$years_of_experience[which(years_of_experience_defense$years_of_experie
years_of_experience_kicking$years_of_experience[which(years_of_experience_kicking$years_of_experie

player_stats_weekly_offense <- player_stats_weekly_offense %>%
  dplyr::left_join(
    years_of_experience_offense[,c("player_id","season","years_of_experience")],
    by = c("player_id","season")
  )

player_stats_weekly_defense <- player_stats_weekly_defense %>%
  dplyr::left_join(
    years_of_experience_offense[,c("player_id","season","years_of_experience")],
    by = c("player_id","season")
  )
```

```
player_stats_weekly_kicking <- player_stats_weekly_kicking %>%
  dplyr::left_join(
    years_of_experience_offense[,c("player_id","season","years_of_experience")],
    by = c("player_id","season")
  )
```

The `player_stats_weekly` objects are in `player-season-week` form. That is, each row should be uniquely identified by the combination of `player_id`, `season`, and `week`. Let's rearrange the data accordingly:

```
player_stats_weekly_offense <- player_stats_weekly_offense %>%
  arrange(player_display_name, player_id, season, week)

player_stats_weekly_defense <- player_stats_weekly_defense %>%
  arrange(player_display_name, player_id, season, week)

player_stats_weekly_kicking <- player_stats_weekly_kicking %>%
  arrange(player_display_name, player_id, season, week)
```

Let's check for duplicate `player-season-week` instances:

```
player_stats_weekly_offense %>%
  group_by(player_id, season, week) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 88
# Groups:   player_id, season, week [0]
# i 88 variables: player_id <chr>, season <int>, week <int>, game_id <chr>,
#   gameday <date>, home_team <chr>, away_team <chr>, birth_date <date>,
#   age <dbl>, ageCentered20 <dbl>, ageCentered20Quadratic <dbl>, status <chr>,
#   display_name <chr>, first_name <chr>, last_name <chr>, esb_id <chr>,
#   college_name <chr>, jersey_number <int>, height <dbl>, weight <int>,
#   current_team_id <chr>, football_name <chr>, entry_year <int>,
#   rookie_year <int>, draft_club <chr>, draft_number <int>, ...
```

```
player_stats_weekly_defense %>%
  group_by(player_id, season, week) %>%
  filter(n() > 1) %>%
  head() #todo: fix duplicate instances
```

```
# A tibble: 6 x 67
# Groups:   player_id, season, week [6]
```

```
  player_id season  week game_id gameday home_team away_team birth_date   age
  <chr>     <int> <int> <chr>   <date>  <chr>    <chr>    <date>    <dbl>
1 00-00328~   2016     1 <NA>    NA      <NA>     <NA>     NA           NA
2 00-00328~   2016     2 <NA>    NA      <NA>     <NA>     NA           NA
3 00-00328~   2016     3 <NA>    NA      <NA>     <NA>     NA           NA
4 00-00328~   2016     4 <NA>    NA      <NA>     <NA>     NA           NA
5 00-00328~   2016     5 <NA>    NA      <NA>     <NA>     NA           NA
6 00-00328~   2016     6 <NA>    NA      <NA>     <NA>     NA           NA
# i 58 more variables: ageCentered20 <dbl>, ageCentered20Quadratic <dbl>,
#   status <chr>, display_name <chr>, first_name <chr>, last_name <chr>,
#   esb_id <chr>, college_name <chr>, jersey_number <int>, height <dbl>,
#  weight <int>, current_team_id <chr>, football_name <chr>, entry_year <int>,
#   rookie_year <int>, draft_club <chr>, draft_number <int>,
#   college_conference <chr>, status_description_abbr <chr>,
#  status_short_description <chr>, gsis_it_id <int>, short_name <chr>, ...
```

```r
player_stats_weekly_kicking %>%
  group_by(player_id, season, week) %>%
  filter(n() > 1) %>%
  head() #todo: fix duplicate instances
```

```
# A tibble: 6 x 79
# Groups:   player_id, season, week [6]
  player_id season  week game_id gameday home_team away_team birth_date   age
  <chr>     <int> <int> <chr>   <date>  <chr>    <chr>    <date>    <dbl>
1 00-00210~   2003     1 <NA>    NA      <NA>     <NA>     NA           NA
2 00-00210~   2003     2 <NA>    NA      <NA>     <NA>     NA           NA
3 00-00210~   2003     3 <NA>    NA      <NA>     <NA>     NA           NA
4 00-00210~   2003     4 <NA>    NA      <NA>     <NA>     NA           NA
5 00-00210~   2003     5 <NA>    NA      <NA>     <NA>     NA           NA
6 00-00210~   2003     7 <NA>    NA      <NA>     <NA>     NA           NA
# i 70 more variables: ageCentered20 <dbl>, ageCentered20Quadratic <dbl>,
#   status <chr>, display_name <chr>, first_name <chr>, last_name <chr>,
#   esb_id <chr>, college_name <chr>, jersey_number <int>, height <dbl>,
#  weight <int>, current_team_id <chr>, football_name <chr>, entry_year <int>,
#   rookie_year <int>, draft_club <chr>, draft_number <int>,
#   college_conference <chr>, status_description_abbr <chr>,
#  status_short_description <chr>, gsis_it_id <int>, short_name <chr>, ...
```

```r
# Save data
save(
  player_stats_weekly_offense, player_stats_weekly_defense, player_stats_weekly_kicking,
  file = "./data/player_stats_weekly.RData"
)
```

4.4.3.2 Seasonal

```
# Merge player info with seasonal stats
player_stats_seasonal_offense <- dplyr::full_join(
  nfl_actualStats_offense_seasonal,
  nfl_players %>% select(-position, -position_group, -season, -team_abbr, - team_seq),
  by = c("player_id" = "gsis_id")
)

player_stats_seasonal_defense <- dplyr::full_join(
  nfl_actualStats_defense_seasonal,
  nfl_players %>% select(-position, -position_group, -season, -team_abbr, - team_seq),
  by = c("player_id" = "gsis_id")
)

player_stats_seasonal_kicking <- dplyr::full_join(
  nfl_actualStats_kicking_seasonal,
  nfl_players %>% select(-position, -position_group, -season, -team_abbr, - team_seq),
  by = c("player_id" = "gsis_id")
)

# Calculate age
season_startdate <- nfl_schedules %>%
  dplyr::group_by(season) %>%
  dplyr::summarise(startdate = min(gameday, na.rm = TRUE))

player_stats_seasonal_offense <- player_stats_seasonal_offense %>%
  dplyr::left_join(
    season_startdate,
    by = "season"
  )

player_stats_seasonal_defense <- player_stats_seasonal_defense %>%
  dplyr::left_join(
    season_startdate,
    by = "season"
  )

player_stats_seasonal_kicking <- player_stats_seasonal_kicking %>%
  dplyr::left_join(
    season_startdate,
    by = "season"
  )
```

```
player_stats_seasonal_offense$age <- lubridate::interval(
  start = player_stats_seasonal_offense$birth_date,
  end = player_stats_seasonal_offense$startdate
) %>%
  lubridate::time_length(unit = "years")

player_stats_seasonal_defense$age <- lubridate::interval(
  start = player_stats_seasonal_defense$birth_date,
  end = player_stats_seasonal_defense$startdate
) %>%
  lubridate::time_length(unit = "years")

player_stats_seasonal_kicking$age <- lubridate::interval(
  start = player_stats_seasonal_kicking$birth_date,
  end = player_stats_seasonal_kicking$startdate
) %>%
  lubridate::time_length(unit = "years")

# Merge with Pro Football Reference Data on Player Age by Season
player_stats_seasonal_offense <- player_stats_seasonal_offense %>%
  dplyr::left_join(
    nfl_advancedStatsPFR_seasonal %>% filter(!is.na(gsis_id), !is.na(season), !is.na(age)) %>% sel
    by = c("player_id" = "gsis_id", "season")
  )

player_stats_seasonal_defense <- player_stats_seasonal_defense %>%
  dplyr::left_join(
    nfl_advancedStatsPFR_seasonal %>% filter(!is.na(gsis_id), !is.na(season), !is.na(age)) %>% sel
    by = c("player_id" = "gsis_id", "season")
  )

player_stats_seasonal_kicking <- player_stats_seasonal_kicking %>%
  dplyr::left_join(
    nfl_advancedStatsPFR_seasonal %>% filter(!is.na(gsis_id), !is.na(season), !is.na(age)) %>% sel
    by = c("player_id" = "gsis_id", "season")
  )

# Set age as first non-missing value from calculation above or from PFR
player_stats_seasonal_offense <- player_stats_seasonal_offense %>%
  mutate(age = coalesce(age.x, age.y)) %>%
  select(-age.x, -age.y)

player_stats_seasonal_defense <- player_stats_seasonal_defense %>%
  mutate(age = coalesce(age.x, age.y)) %>%
```

```
  select(-age.x, -age.y)

player_stats_seasonal_kicking <- player_stats_seasonal_kicking %>%
  mutate(age = coalesce(age.x, age.y)) %>%
  select(-age.x, -age.y)

# Calculate ageCentered and ageCenteredQuadratic
player_stats_seasonal_offense$ageCentered20 <- player_stats_seasonal_offense$age - 20
player_stats_seasonal_offense$ageCentered20Quadratic <- player_stats_seasonal_offense$ageCentered2

player_stats_seasonal_defense$ageCentered20 <- player_stats_seasonal_defense$age - 20
player_stats_seasonal_defense$ageCentered20Quadratic <- player_stats_seasonal_defense$ageCentered2

player_stats_seasonal_kicking$ageCentered20 <- player_stats_seasonal_kicking$age - 20
player_stats_seasonal_kicking$ageCentered20Quadratic <- player_stats_seasonal_kicking$ageCentered2

# Years of experience
player_stats_seasonal_offense$years_of_experience <- NULL
player_stats_seasonal_defense$years_of_experience <- NULL
player_stats_seasonal_kicking$years_of_experience <- NULL

player_stats_seasonal_offense <- player_stats_seasonal_offense %>%
  dplyr::left_join(
    years_of_experience_offense[,c("player_id","season","years_of_experience")],
    by = c("player_id","season")
  )

player_stats_seasonal_defense <- player_stats_seasonal_defense %>%
  dplyr::left_join(
    years_of_experience_offense[,c("player_id","season","years_of_experience")],
    by = c("player_id","season")
  )

player_stats_seasonal_kicking <- player_stats_seasonal_kicking %>%
  dplyr::left_join(
    years_of_experience_offense[,c("player_id","season","years_of_experience")],
    by = c("player_id","season")
  )
```

The `player_stats_seasonal` objects are in `player-season` form. That is, each row should be uniquely identified by the combination of `player_id` and `season`. Let's rearrange the data accordingly:

```
player_stats_seasonal_offense <- player_stats_seasonal_offense %>%
  select(player_id, season, everything()) %>%
  arrange(player_display_name, player_id, season)

player_stats_seasonal_defense <- player_stats_seasonal_defense %>%
  select(player_id, season, everything()) %>%
  arrange(player_display_name, player_id, season)

player_stats_seasonal_kicking <- player_stats_seasonal_kicking %>%
  select(player_id, season, everything()) %>%
  arrange(player_display_name, player_id, season)
```

Let's check for duplicate `player-season` instances:

```
player_stats_seasonal_offense %>%
  group_by(player_id, season) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 82
# Groups:   player_id, season [0]
# i 82 variables: player_id <chr>, season <int>, player_name <chr>,
#   player_display_name <chr>, position <chr>, position_group <chr>,
#   headshot_url <chr>, games <int>, team <chr>, completions <int>,
#   attempts <int>, passing_yards <dbl>, passing_tds <int>,
#  interceptions <dbl>, sacks <dbl>, sack_yards <dbl>, sack_fumbles <int>,
#   sack_fumbles_lost <int>, passing_air_yards <dbl>,
#   passing_yards_after_catch <dbl>, passing_first_downs <dbl>, ...
```

```
player_stats_seasonal_defense %>%
  group_by(player_id, season) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 62
# Groups:   player_id, season [0]
# i 62 variables: player_id <chr>, season <int>, player_name <chr>,
#   player_display_name <chr>, position <chr>, position_group <chr>,
#   team <chr>, headshot_url <chr>, games <int>, def_tackles <int>,
#   def_tackles_solo <int>, def_tackles_with_assist <int>,
#   def_tackle_assists <int>, def_tackles_for_loss <int>,
#   def_tackles_for_loss_yards <dbl>, def_fumbles_forced <int>,
#   def_sacks <dbl>, def_sack_yards <dbl>, def_qb_hits <dbl>, ...
```

```r
player_stats_seasonal_kicking %>%
  group_by(player_id, season) %>%
  filter(n() > 1) %>%
  head()
```

```
# A tibble: 0 x 70
# Groups:   player_id, season [0]
# i 70 variables: player_id <chr>, season <int>, player_name <chr>,
#   player_display_name <chr>, position <chr>, position_group <chr>,
#  team <chr>, headshot_url <chr>, games <int>, fg_made <int>, fg_att <dbl>,
#   fg_missed <int>, fg_blocked <int>, fg_long <dbl>, fg_pct <dbl>,
#   fg_made_0_19 <int>, fg_made_20_29 <int>, fg_made_30_39 <int>,
#   fg_made_40_49 <int>, fg_made_50_59 <int>, fg_made_60_ <int>,
#  fg_missed_0_19 <int>, fg_missed_20_29 <int>, fg_missed_30_39 <int>, ...
```

```r
# Save data
save(
  player_stats_seasonal_offense, player_stats_seasonal_defense, player_stats_seasonal_kicking,
  file = "./data/player_stats_seasonal.RData"
)
```

5

Data Visualization

5.1 Getting Started

5.1.1 Load Packages

```
library("nflplotR")
library("plotly")
library("gghighlight")
library("tidyverse")
```

5.1.2 Load Data

```
load(file = "./data/nfl_pbp.RData")
load(file = "./data/player_stats_weekly.RData")
load(file = "./data/player_stats_seasonal.RData")
```

We created the `player_stats_weekly.RData` and `player_stats_seasonal.RData` objects in Section 4.4.3.

5.2 Overview

5.2.1 Principles of Graphic Design

When designing graphics, it is important to understand general principles of graphic design. An article that describes important principles about graphic design is at the following link: https://www.adobe.com/express/learn/blog/8-

basic-design-principles-to-help-you-create-better-graphics (archived at https://perma.cc/29P9-NNSK). The important principles include:

---

1. Focus on alignment.
2. Use hierarchy to help focus your design.
3. Leverage contrast to accentuate important design elements.
4. Use repetition to your advantage.
5. Consider proximity when organizing your graphic elements.
6. Make sure your designs have balance.
7. Optimize color to support your design.
8. Leave negative space.

5.2.2 Principles of Data Visualization

Data visualization involves graphic design in a particular domain—the visualization of data (numeric-derived information). Schwabish (2021) describes five principles in data visualization:

---

1. Show the data.
2. Reduce the clutter.
3. Integrate the graphics and text.
4. Avoid the spaghetti chart.
5. Start with gray.

In his classic book, Tufte (2001) states that effective data visualizations should follow principles of graphical excellence and integrity. He notes that "Graphical excellence is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space." (p. 51). That is, data visualizations should seek to maximize the data-to-ink ratio (within reason),

and should spend less space on "fluff" (i.e., non-data things that can be erased without losing meaning, such as grid lines, redundancies, etc.). Tufte (2001) describes six principles of graphical integrity:

---

1. The representation of numbers, as physically measured on the surface of the graphic itself, should be directly proportional to the numerical quantities represented.
2. Clear, detailed, and thorough labeling should be used to defeat graphic distortion and ambiguity. Write out explanations of the data on the graphic itself. Label important events in the data.
3. Show data variation, not design variation.
4. In time-series displays of money, deflated and standardized units of monetary measurement are nearly always better than nominal units.
5. The number of information-carrying (variable) dimensions depicted should not exceed the number of dimensions in the data.
6. Graphics must not quote data out of context.

— Tufte (2001, p. 77)

Tufte (2001) also provides recommendations for friendly, accessible graphics, including:

- spell words out (rather than using abbreviations)
- have words run from left to write (including the y-axis title)
- include little messages to help explain the data
- place labels on the graphic so no legend is needed
- avoid elaborately encoded shadings, cross-hatching, and colors
- avoid "chartjunk"—i.e., unnecessary or distracting elements (e.g., excessive decoration, overly complex graphics, graphical effects, and irrelevant information such as moiré vibration, heavy grids, and self-promoting graphs) that do not improve viewers' understanding of the data
- if colors are used, use colors that are distinguishable by color-deficient and color-blind viewers (red–green is a common form of color-blindness)
- use type (i.e., of the text) that is clear, precise, and modest
- use text that is upper-and-lower case, not all capitals

5.2.3 Creating Data Visualizations in R

The R Graph Gallery provides examples of various types of plots: https://r-graph-gallery.com. Books on data visualization in R include *ggplot2: Elegant Graphics for Data Analysis*[1] (Wickham, 2024) and *R Graphics Cookbook: Practical Recipes for Visualizing Data*[2] (Chang, 2018). In this chapter, we will examine how to create statistical graphics to visualize data. We will create the plots using the `ggplot2` package. When creating plots in `ggplot2` with multiple points or lines (e.g., multiple players or levels of a predictor variable), it is easiest to do so with the data in long form (as opposed to wide form).

A key principle of graphic design and data visualization is the importance of contrast. Each visual component (e.g., line) that is important to see should be easy to distinguish. For instance, you can highlight lines or points of interest to draw people's attention to the target of interest. For examples of highlighting in figures, see Figures 5.17 (Section 5.5.1) and 14.3. It is also important to use color schemes with distinguishable colors. Good color schemes for sequential, diverging, and qualitative (i.e., categorical) data are provided by ColorBrewer (https://colorbrewer2.org) and are available using the `scale_color_brewer()` and `scale_fill_brewer()` functions of the `ggplot2` package, as demonstrated in Figure 5.20 (Section 5.6.2).

5.3 Univariate Distribution

5.3.1 Histogram

A histogram of fantasy points is depicted in Figure 5.1.

```
ggplot2::ggplot(
  data = player_stats_seasonal_offense %>%
    filter(position_group %in% c("QB","RB","WR","TE")),
  mapping = aes(
    x = fantasy_points)
) +
  geom_histogram(
    color = "#000000",
    fill = "#0099F8"
  ) +
  labs(
```

[1] https://ggplot2-book.org
[2] https://r-graphics.org

```
    x = "Fantasy Points",
    title = "Histogram of Fantasy Points"
  ) +
  theme_classic()
```

Figure 5.1 Histogram of Fantasy Points.

5.3.2 Density Plot

A histogram of fantasy points is depicted in Figure 5.2.

```
ggplot2::ggplot(
  data = player_stats_seasonal_offense %>%
    filter(position_group %in% c("QB","RB","WR","TE")),
  mapping = aes(
    x = fantasy_points,
    fill = position_group)
) +
  geom_density(alpha = 0.7) + # add transparency
  labs(
    x = "Fantasy Points",
    fill = "Position",
    title = "Density Plot of Fantasy Points by Position"
  ) +
```

```
  theme_classic()
```

Figure 5.2 Density Plot of Fantasy Points by Position.

5.3.3 Histogram with Overlaid Density and Rug Plot

A histogram of fantasy points with an overlaid density and rug plot is depicted in Figure 5.3.

```
ggplot2::ggplot(
  data = player_stats_seasonal_offense %>%
    filter(position_group %in% c("QB","RB","WR","TE")),
  mapping = aes(
    x = fantasy_points)
) +
  geom_histogram(
    aes(y = after_stat(density)),
    color = "#000000",
    fill = "#0099F8"
  ) +
  geom_density(
    color = "#000000",
    fill = "#F85700",
    alpha = 0.6 # add transparency
```

```
  ) +
  geom_rug() +
  labs(
    x = "Fantasy Points",
    title = "Histogram of Fantasy Points with Overlaid Density and Rug Plot"
  ) +
  theme_classic()
```

Figure 5.3 Histogram with Overlaid Density and Rug Plot.

5.3.4 Box-and-Whisker Plot

A box-and-whisker plot of fantasy points is depicted in Figure 5.4.

```
ggplot2::ggplot(
  data = player_stats_seasonal_offense %>%
    filter(position_group %in% c("QB","RB","WR","TE")),
  mapping = aes(
    x = position_group,
    y = fantasy_points,
    fill = position_group)
) +
  geom_boxplot() +
  labs(
```

```
    x = "Position",
    y = "Fantasy Points",
    title = "Box-and-Whisker Plot of Fantasy Points by Position"
  ) +
  theme_classic() +
  theme(legend.position = "none")
```

Figure 5.4 Box-and-Whisker Plot.

5.3.5 Violin Plot

A violin plot of fantasy points is depicted in Figure 5.4.

```
ggplot2::ggplot(
  data = player_stats_seasonal_offense %>%
    filter(position_group %in% c("QB","RB","WR","TE")),
  mapping = aes(
    x = position_group,
    y = fantasy_points,
    fill = position_group)
) +
  geom_violin(draw_quantiles = c(0.25, 0.5, 0.75)) +
  labs(
    x = "Position",
```

```
    y = "Fantasy Points",
    title = "Violin Plot by Position",
    subtitle = "Lines represent the 25th, 50th, and 75th quantiles"
  ) +
  theme_classic() +
  theme(legend.position = "none")
```

Figure 5.5 Violin Plot. Lines represent the 25th, 50th, and 75th quantiles.

5.4 Scatterplot

As a tutorial, we walk through some of the (many) modifications that can be made to create an advanced, customized plot in `ggplot2`.

First, we prepare the data:

```
# Subset Data
rb_seasonal <- player_stats_seasonal_offense %>%
  filter(position_group == "RB")
```

5.4.1 Base Layer

Second, we create the base layer of the plot using the `ggplot()` function of the `ggplot2` package, as in Figure 5.6. We specify the data object and the variables in the data object that are associated with the x- and y-axes:

```
ggplot2::ggplot(
  data = rb_seasonal, # specify data object
  aes(
    x = age, # specify variable on x-axis
    y = rushing_yards)) # specify variable on y-axis
```

Figure 5.6 Base Plot.

5.4.2 Add Points

Third, we create a scatterplot using the `geom_point()` function from the `ggplot2` package, as in Figure 5.7:

```
ggplot2::ggplot(
  data = rb_seasonal,
  aes(
    x = age,
    y = rushing_yards)) +
  geom_point() # add points for scatterplot
```

Figure 5.7 Scatterplot.

5.4.3 Best-Fit Line

Fourth, we add a linear best-fit line using the `geom_smooth()`, as in Figure 5.8:

```
ggplot2::ggplot(
  data = rb_seasonal,
  aes(
    x = age,
    y = rushing_yards)) +
  geom_point() +
  geom_smooth(method = "lm") # add linear best-fit line
```

Figure 5.8 Scatterplot with Linear Best-Fit Line.

We could also estimate a quadratic polynomial best-fit line, as in Figure 5.9:

```
ggplot2::ggplot(
  data = rb_seasonal,
  aes(
    x = age,
    y = rushing_yards)) +
  geom_point() +
  geom_smooth(
    method = "lm",
    formula = y ~ poly(x, 2)) # add quadratic best-fit line
```

Figure 5.9 Scatterplot with Quadratic Best-Fit Line.

Or, we could estimate a smooth best-fit line using locally estimated scatterplot smoothing (LOESS) to allow for any form of nonlinearity, as in Figure 5.10:

```
ggplot2::ggplot(
  data = rb_seasonal,
  aes(
    x = age,
    y = rushing_yards)) +
  geom_point() +
  geom_smooth(method = "loess") # add smooth best-fit (LOESS) line
```

Figure 5.10 Scatterplot with Best-Fit Line Using Locally Estimated Scatterplot Smoothing (LOESS).

By default, the best-fit line is based on a generalized additive model, which allows for nonlinearity, as in Figure 5.11:

```
ggplot2::ggplot(
  data = rb_seasonal,
  aes(
    x = age,
    y = rushing_yards)) +
  geom_point() +
  geom_smooth() # add GAM best-fit line; same as specifying method = "gam"
```

Figure 5.11 Scatterplot with Best-Fit Line from Generalized Additive Model.

5.4.4 Modify Axes

Then, we can change the axes, as in Figure 5.12:

```
ggplot2::ggplot(
  data = rb_seasonal,
  aes(
    x = age,
    y = rushing_yards)) +
  geom_point() +
  geom_smooth() +
  scale_x_continuous(
    expand = c(0,0), # set origin of x-axis to 0
    lim = c(20,40), # set limits of x-axis
    breaks = seq(from = 20, to = 40, by = 5) # specify x-axis labels
  ) +
  scale_y_continuous(
    expand = c(0,0), # set origin of y-axis to 0
    lim = c(0,NA), # set limits of y-axis
    breaks = seq(from = 0, to = 2500, by = 250) # specify y-axis labels
  )
```

Figure 5.12 Scatterplot with Modified Axes.

5.4.5 Plot Labels

Then, we can add plot labels, as in Figure 5.13:

```
ggplot2::ggplot(
  data = rb_seasonal,
  aes(
    x = age,
    y = rushing_yards)) +
  geom_point() +
  geom_smooth() +
  scale_x_continuous(
    expand = c(0,0),
    lim = c(20,40),
    breaks = seq(from = 20, to = 40, by = 5)
  ) +
  scale_y_continuous(
    expand = c(0,0),
    lim = c(0,NA),
    breaks = seq(from = 0, to = 2500, by = 250)
  ) +
  labs( # add plot labels
    x = "Running Back's Age (years)",
```

```
    y = "Rushing Yards (Season)",
    title = "NFL Rushing Yards (Season) by Player Age",
    subtitle = "(Among Running Backs)"
  )
```

Figure 5.13 Scatterplot with Plot Labels.

5.4.6 Theme

Then, we can use a theme such as the classic theme (`theme_classic()`) to make it more visually presentable, as in Figure 5.14:

```
ggplot2::ggplot(
  data = rb_seasonal,
  aes(
    x = age,
    y = rushing_yards)) +
  geom_point() +
  geom_smooth() +
  scale_x_continuous(
    expand = c(0,0),
    lim = c(20,40),
    breaks = seq(from = 20, to = 40, by = 5)
  ) +
```

```
  scale_y_continuous(
    expand = c(0,0),
    lim = c(0,NA),
    breaks = seq(from = 0, to = 2500, by = 250)
  ) +
  labs(
    x = "Running Back's Age (years)",
    y = "Rushing Yards (Season)",
    title = "NFL Rushing Yards (Season) by Player Age",
    subtitle = "(Among Running Backs)"
  ) +
  theme_classic() # use the classic theme
```

Figure 5.14 Scatterplot with Classic Theme.

Or, we could use a different theme, such as the dark theme (`theme_dark()`) in Figure 5.15. For a list of themes available in `ggplot2`, see here: https://ggplot2-book.org/themes#sec-themes.

```
ggplot2::ggplot(
  data = rb_seasonal,
  aes(
    x = age,
    y = rushing_yards)) +
  geom_point() +
```

```
  geom_smooth() +
  scale_x_continuous(
    expand = c(0,0),
    lim = c(20,40),
    breaks = seq(from = 20, to = 40, by = 5)
  ) +
  scale_y_continuous(
    expand = c(0,0),
    lim = c(0,NA),
    breaks = seq(from = 0, to = 2500, by = 250)
  ) +
  labs(
    x = "Running Back's Age (years)",
    y = "Rushing Yards (Season)",
    title = "NFL Rushing Yards (Season) by Player Age",
    subtitle = "(Among Running Backs)"
  ) +
  theme_dark() # use the dark theme
```

Figure 5.15 Scatterplot with Dark Theme.

5.4.7 Interactive

After creating our plot, we can make the plot interactive using the `ggplotly()` function from the `plotly` package.

```
plot_ypcByPlayerAge <- ggplot2::ggplot(
  data = rb_seasonal,
  aes(
    x = age,
    y = rushing_yards)) +
  geom_point(
    aes(
      text = player_display_name, # add player name for mouse over tooltip
      label = season)) + # add season for mouse over tooltip
  geom_smooth() +
  scale_x_continuous(
    expand = c(0,0),
    lim = c(20,40),
    breaks = seq(from = 20, to = 40, by = 5)
  ) +
  scale_y_continuous(
    expand = c(0,0),
    lim = c(0,NA),
    breaks = seq(from = 0, to = 2500, by = 250)
  ) +
  labs(
    x = "Running Back's Age (years)",
    y = "Rushing Yards (Season)",
    title = "NFL Rushing Yards (Season) by Player Age",
    subtitle = "(Among Running Backs)"
  ) +
  theme_classic()

ggplotly(plot_ypcByPlayerAge)
```

5.5 Line Chart

A bar plot of Tom Brady's fantasy points by season is depicted in Figure 5.16.

```
ggplot2::ggplot(
  data = player_stats_seasonal_offense %>%
    filter(player_display_name == "Tom Brady"),
  mapping = aes(
    x = season,
    y = fantasy_points
    )
) +
  geom_line(
    linewidth = 1.5,
    color = "blue"
  ) +
  labs(
    x = "Season",
    y = "Fantasy Points",
    title = "Bar Plot of Tom Brady's Fantasy Points by Season"
  ) +
  theme_classic()
```

Figure 5.16 Line Chart.

5.5.1 With Highlighting

```
ggplot2::ggplot(
  data = player_stats_seasonal_offense,
  mapping = aes(
    x = season,
    y = fantasy_points,
    group = player_id,
    color = player_display_name)
) +
  geom_line() +
  gghighlight::gghighlight(
    player_display_name == "Tom Brady",
    label_key = player_display_name) +
  labs(
    x = "Season",
    y = "Fantasy Points",
    title = "Fantasy Points by Season and Player",
    subtitle = "(Tom Brady in Red)"
  ) +
  theme_classic()
```

Figure 5.17 Line Chart with Highlighting.

Table 5.1 Table of Summary Statistics.

```
player_stats_seasonal_offense_summary
```

```
# A tibble: 4 x 7
  position_group     n  mean    sd    se ci_lower ci_upper
  <chr>          <int> <dbl> <dbl> <dbl>    <dbl>    <dbl>
1 QB              1929 105.  114.  2.59      99.5    110.
2 RB              4043  61.2  72.9 1.15      59.0     63.4
3 TE              2693  33.6  39.3 0.756     32.1     35.1
4 WR              4867  58.3  58.9 0.845     56.7     60.0
```

5.6 Bar Plot

To create a bar plot, we first compute summary statistics:

```
confidenceLevel <- .95 # for 95% confidence interval

player_stats_seasonal_offense_summary <- player_stats_seasonal_offense %>%
    filter(position_group %in% c("QB","RB","WR","TE")) %>%
  group_by(position_group) %>%
  summarise(
    n = sum(!is.na(fantasy_points)),
    mean = mean(fantasy_points, na.rm = TRUE),
    sd = sd(fantasy_points, na.rm = TRUE)
  ) %>%
  mutate(se = sd/sqrt(n)) %>%
  mutate(
    ci_lower = mean - qt(p = 1 - (1 - confidenceLevel) / 2, df = n - 1) * se,
    ci_upper = mean + qt(p = 1 - (1 - confidenceLevel) / 2, df = n - 1) * se
  )
```

The summary statistics are in Table 5.1.

A bar plot of fantasy points by position is depicted in Figure 5.18.

```
ggplot2::ggplot(
  data = player_stats_seasonal_offense_summary,
  mapping = aes(
    x = position_group,
    y = mean,
```

```
    fill = position_group,
    color = position_group
    )
) +
  geom_bar(
    stat = "identity") +
  labs(
    x = "Position",
    y = "Fantasy Points",
    title = "Bar Plot of Fantasy Points by Position"
  ) +
  theme_classic() +
  theme(legend.position = "none")
```

Figure 5.18 Bar Plot.

5.6.1 With Error Bars

Based on the summary statistics in Table 5.1, we create a bar plot with bars representing the 95% confidence interval in Figure 5.19.

```
ggplot2::ggplot(
  data = player_stats_seasonal_offense_summary %>%
    filter(position_group %in% c("QB","RB","WR","TE")),
```

```
  mapping = aes(
    x = position_group,
    y = mean,
    fill = position_group,
    color = position_group
    )
) +
  geom_bar(
    stat = "identity") +
  geom_errorbar(
    aes(
      ymin = ci_lower,
      ymax = ci_upper),
    width = 0.2,
    color = "black"
  ) +
  labs(
    x = "Position",
    y = "Fantasy Points",
    title = "Bar Plot of Fantasy Points by Position"
  ) +
  theme_classic() +
  theme(legend.position = "none")
```

Figure 5.19 Bar Plot with Bars Representing the 95% Confidence Interval.

5.6.2 Modified Color Scheme

We can also modify the color scheme, as in Figure 5.20

```
ggplot2::ggplot(
  data = player_stats_seasonal_offense_summary %>%
    filter(position_group %in% c("QB","RB","WR","TE")),
  mapping = aes(
    x = position_group,
    y = mean,
    fill = position_group,
    color = position_group
    )
) +
  geom_bar(
    stat = "identity") +
  scale_fill_brewer(palette = "Set1") +
  geom_errorbar(
    aes(
      ymin = ci_lower,
      ymax = ci_upper),
    width = 0.2,
    color = "black"
  ) +
  labs(
    x = "Position",
    y = "Fantasy Points",
    title = "Bar Plot of Fantasy Points by Position"
  ) +
  theme_classic() +
  theme(legend.position = "none")
```

Figure 5.20 Bar Plot with Bars Representing the 95% Confidence Interval.

5.7 Examples

5.7.1 Players

5.7.1.1 Running Back Performance By Player Age

```
# Prepare Data
rushing_attempts <- nfl_pbp %>%
  dplyr::filter(season_type == "REG") %>%
  dplyr::filter(
    rush == 1,
    rush_attempt == 1,
    qb_scramble == 0,
    qb_dropback == 0,
    !is.na(rushing_yards))

rb_yardsPerCarry <- rushing_attempts %>%
  dplyr::group_by(rusher_id, season) %>%
  dplyr::summarise(
```

```
    ypc = mean(rushing_yards, na.rm = TRUE),
    rush_attempts = n(),
    .groups = "drop") %>%
  dplyr::ungroup() %>%
  dplyr::left_join(
    player_stats_seasonal_offense,
    by = c("rusher_id" = "player_id", "season")
  ) %>%
  dplyr::filter(
    position_group == "RB",
    rush_attempts >= 50)
```

5.7.1.1.1 Rushing Yards Per Carry

Rushing yards per carry over the course of the season is depicted as a function of the Running Back's age in Figure 5.21.

```
ggplot2::ggplot(
  data = rb_yardsPerCarry,
  aes(
    x = age,
    y = ypc)) +
  geom_point(
    aes(
      text = player_display_name,
      label = season)) +
  geom_smooth() +
  labs(
    x = "Running Back's Age (years)",
    y = "Rushing Yards Per Carry (Season)",
    title = "NFL Rushing Yards Per Carry (Season) by Player Age",
    subtitle = "(minimum 50 rushing attempts)"
  ) +
  theme_classic()
```

Figure 5.21 NFL Rushing Yards Per Carry Per Season by Player Age.

5.7.1.1.2 Rushing EPA Per Season

Rushing expected points added (EPA) over the course of the season is depicted as a function of the Running Back's age in Figure 5.22.

```
ggplot2::ggplot(
  data = rb_seasonal,
  aes(
    x = age,
    y = rushing_epa)) +
  geom_point(
    aes(
      text = player_display_name,
      label = season)) +
  geom_smooth() +
  labs(
    x = "Running Back's Age (years)",
    y = "Rushing EPA (Season)",
    title = "NFL Rushing Expected Points Added (Season) by Player Age"
  ) +
  theme_classic()
```

Figure 5.22 NFL Rushing Expected Points Added (EPA) During a Season by Player Age.

5.7.2 Teams

5.7.2.1 Defensive and Offensive EPA per Play

Expected points added (EPA) per play by the team with possession.

```
pbp_regularSeason <- nfl_pbp %>%
  dplyr::filter(
    season == 2023,
    season_type == "REG") %>%
  dplyr::filter(!is.na(posteam) & (rush == 1 | pass == 1))

epa_offense <- pbp_regularSeason %>%
  dplyr::group_by(team = posteam) %>%
  dplyr::summarise(off_epa = mean(epa, na.rm = TRUE))

epa_defense <- pbp_regularSeason %>%
  dplyr::group_by(team = defteam) %>%
  dplyr::summarise(def_epa = mean(epa, na.rm = TRUE))

epa_combined <- epa_offense %>%
  dplyr::inner_join(
```

```
    epa_defense,
    by = "team")
```

Defensive EPA per play during the 2023 NFL season is depicted as a function of offensive EPA per play in Figure 5.23.

```
ggplot2::ggplot(
  data = epa_combined,
  aes(
    x = off_epa,
    y = def_epa)) +
  nflplotR::geom_mean_lines(
    aes(
      x0 = off_epa ,
      y0 = def_epa)) +
  nflplotR::geom_nfl_logos(
    aes(
      team_abbr = team),
      width = 0.065,
      alpha = 0.7) +
  labs(
    x = "Offense EPA/play",
    y = "Defense EPA/play",
    title = "2023 NFL Offensive and Defensive EPA per Play"
  ) +
  theme_classic() +
  scale_y_reverse()
```

Figure 5.23 2023 NFL Offensive and Defensive EPA Per Play.

5.8 Conclusion

6

Player Evaluation

6.1 Getting Started

6.1.1 Load Packages

```
library("tidyverse")
```

6.2 Overview

Evaluating players for fantasy football could be thought of as similar to the process of evaluating companies when picking stocks to buy. You want to evaluate and compare various assets so that you get the assets with the best value.

There are various domains of criteria we can consider when evaluating a football player's fantasy prospects. Potential domains to consider include:

- athletic profile
- historical performance
- health
- age and career stage
- situational factors
- matchups
- cognitive and motivational factors
- fantasy value

The discussion that follows is based on my and others' *impressions* of some of the characteristics that may be valuable to consider when evaluating players. However, the extent to which any factor is actually relevant for predicting

future performance is an empirical question and should be evaluated empirically.

6.3 Athletic Profile

Factors related to a player's athletic profile include factors such as:

- body shape
  - height
  - weight
  - hand size
  - wing span (arm length)
- body function
  - agility
  - strength
  - speed
  - acceleration/explosiveness
  - jumping ability

In terms of body shape, we might consider a player's height, weight, hand size, and wing span (arm length). Height allows players to see over opponents and to reach balls higher in the air. Thus, greater height is particularly valuable for Quarterbacks and Wide Receivers. Heavier players are tougher to budge and to tackle. Greater weight is particularly valuable for Linemen, Fullbacks, and Tight Ends, but it can also be valuable—to a deree—for Quarterbacks, Running Backs, and Wide Receivers. Hand size and wing span is particularly valuable for people catching the ball; thus, a larger hand size and longer wing span are particularly valuable for Wide Receivers and Tight Ends.

In terms of body function, we can consider a player's agility, strength, speed, acceleration/explosiveness, and jumping ability. For Wide Receivers, speed, explosiveness, and jumping ability are particularly valuable. For Running Backs, agility, strength, speed, and explosiveness are particularly valuable.

Many aspects of a player's athletic profile, including tests of speed (40-yard dash), strength (bench press), agility (20-yard shuttle run; three cone drill), and jumping ability (vertical jump; broad jump) are available from the National Football League (NFL) Combine, which is especially relevant for evaluating rookies. We demonstrate how to import data from the NFL Combine in Section 4.3.8. There are also calculators that integrate information about body shape and information from the NFL Combine to determine a player's relative athletic score (RAS) for their position: https://ras.football/ras-calculator/

6.4 Skill

When scouting players, scouts consider not only the player's athletic profile, but also their position-relevant skill. For instance, how good are they are reading the defense, passing the ball, running routes, catching balls, making defenders miss tackles, taking care of the ball, consistency, etc. Scouting and evaluating skill is a complicated endeavor, and even the professional scouts frequently make mistakes in their evaluations and predictions. You can certainly read skill evaluations about various players; however, unlike metrics of athletic profile, we do not have direct access to the player's underlying skill. Some may say, "You know it when you see it." But, this is not particularly useful when trying to identify players who are undervalued or overvalued—because the skill evaluations are likely already "baked into" a player's projections. Because we do not have direct access to a player's skill, we tend to rely on indirect metrics of their ability, such as historical performance.

6.5 Historical Performance

6.5.1 Overview

> "The best predictor of future behavior is past behavior." – Unknown

> "Past performance does not guarantee future results." – A common disclaimer about investments.

Factors relating to historical performance to consider could include:

- performance in college

  - draft position

- performance in the NFL
- efficiency
- consistency

It is important to consider a player's past performance. However, the extent to which historical performance may predict future performance may depend on many factors such as (a) the similarity of the prior situation to the current situation, (b) how long ago the prior situation was, and (c) the extent to which the player (or situation) has changed in the interim. For rookies, the player does not have prior seasons of performance in the NFL to draw upon. Thus, when evaluating rookies, it can be helpful to consider their performance in college or in their prior leagues. However, there are large differences between the situation in college and the situation in the NFL, so prior success in college may not portend future success in the NFL. An indicator that intends to be prognostic of future performance, and that accounts for past performance, is a player's draft position—that is, how early (or late) was a player selected in the NFL Draft. The earlier a player was selected in the NFL Draft, the greater likelihood that the player will perform well; however, this is somewhat countered by the fact that the teams with the highest draft picks tend to be the worst based on the prior season's record.

For players who have played in the NFL, past performance becomes more relevant because, presumably, the prior situation is more similar (than was their situation in college) to their current situation. Nevertheless, lots of things change from game to game and season to season: injuries, coaches, coaching strategies, teammates, etc. So just because a player performed well or poorly in a given game or season does not necessarily mean that they will perform similarly in subsequent games/seasons. Nevertheless, historical performance is one of the best indicators we have.

We demonstrate how to import historical player statistics in Section 4.3.16. We demonstrate how to calculate historical player statistics in Section 4.4.1. We demonstrate how to calculate historical fantasy points in Section 4.4.2.

6.5.2 Efficiency

In addition to how many fantasy points a player scores in terms of historical performance, we also care about efficiency and consistency. How efficient were they given the number of opportunities they had? This is important to consider because different players have different opportunities. For example, some Running Backs had more opportunity than others (e.g., more carries), so comparing players merely on rushing yards would be misleading. If they were relatively more efficient, they will likely score more points than

many of their peers when given more opportunities. If they were relativelly inefficient, their capacity to score fantasy points may be more dependent on touches/opportunities. Efficiency might be operationalized by indicators such as yards per passing attempt, yards per rushing attempt, yards per target, yards per reception, etc.

6.5.3 Consistency

In terms of consistency, how consistent was the player they from game to game and from season to season? For instance, we could examine the standard deviations of players' fantasy points across games in a given season. However, the standard deviation tends to be upwardly biased as the mean increases. So, we can account for the player's mean fantasy points per game by dividing their game-to-game standard deviation of fantasy points (σ) by their mean fantasy points across games (μ). This is known as the coefficient of variation (CV), which is provided in Equation 6.1.

$$CV = \frac{\sigma}{\mu} \tag{6.1}$$

Players with a lower standard deviation and a lower coefficient of variation (of fantasy points across games) are more consistent. In the example below, Player 2 might be preferable to Player 1 because Player 2 is more consistent; Player 1 is more "boom-or-bust." Despite showing a similar mean of fantasy points across weeks, Player 2 shows a smaller week-to-week standard deviation and coefficient of variation.

```
set.seed(1)

playerScoresByWeek <- data.frame(
  player1_scores = rnorm(17, mean = 20, sd = 7),
  player2_scores = rnorm(17, mean = 20, sd = 4),
  player3_scores = rnorm(17, mean = 10, sd = 4),
  player4_scores = rnorm(17, mean = 10, sd = 1)
)

consistencyData <- data.frame(t(playerScoresByWeek))

weekNames <- paste("week", 1:17, sep = "")

names(consistencyData) <- weekNames
row.names(consistencyData) <- NULL

consistencyData$mean <- rowMeans(consistencyData[,weekNames])
consistencyData$sd <- apply(consistencyData, 1, sd)
```

```
consistencyData$cv <- consistencyData$sd / consistencyData$mean

consistencyData$player <- c(1, 2, 3, 4)

consistencyData <- consistencyData %>%
  select(player, mean, sd, cv, week1:week17)

round(consistencyData, 2)
```

```
  player  mean   sd   cv week1 week2 week3 week4 week5 week6 week7 week8 week9
1      1 20.60 6.47 0.31 15.61 21.29 14.15 31.17 22.31 14.26 23.41 25.17 24.03
2      2 20.61 3.35 0.16 23.78 23.28 22.38 23.68 23.13 20.30 12.04 22.48 19.78
3      3 10.32 2.65 0.26  4.49  8.34  8.42  9.76 14.40 13.05  9.34  8.99 12.79
4      4 10.19 1.11 0.11  9.39 10.34  8.87 11.43 11.98  9.63  8.96 10.57  9.86
  week10 week11 week12 week13 week14 week15 week16 week17
1  17.86  30.58  22.73  15.65   4.50  27.87  19.69  19.89
2  19.38  14.12  18.09  21.67  25.43  19.59  21.55  19.78
3  12.23   7.24   7.17  11.46  13.07   9.55  13.52  11.59
4  12.40   9.96  10.69  10.03   9.26  10.19   8.20  11.47
```

6.6 Health

Health-related factors to consider include:

- current injury status
- injury history

It is also important to consider a player's past and current health status. In terms of a player's current health status, it is important to consider whether they are injured or are playing at less than 100% of their typical health. In terms of a player's prior health status, one can consider their injury history, including the frequency and severity of injuries and their prognosis.

We demonstrate how to import injury reports in Section 4.3.17.

6.7 Availability

In addition to injuries, other factors can affect a player's availability, including league suspensions, for instance, due to violating the league's conduct policy.

6.8 Age and Career Stage

Age and career stage-related factors include:

- age
- experience
- touches

A player's age is relevant because of important age-related changes in a player's speed, ability to recover from injury, etc. A player's experience is relevant because players develop knowledge and skills with greater experience. A player's prior touches/usage is also relevant, because it speaks to how many hits a player may have taken. For players who take more hits, it may be more likely that their bodies "break down" sooner.

6.9 Situational Factors

Situational factors one could consider include:

- team quality
- role on team
- teammates
- opportunity and usage
  - snap count
  - touches/targets
  - red zone usage

Football is a team sport. A player is embedded within a broader team context; it is important to consider the strength of their team context insofar as it

may support— or detract from—a player's performance. For instance, for a Quarterback, it is important to consider how strong the pass blocking is from the Offensive Line. Will they have enough time to throw the ball, or will they be constantly under pressure to be sacked? It is also important to consider the strength of the pass catchers—the Wide Receivers and Tight Ends. For a Running Back, it is important to consider how strong the run blocking is from the Offensive Line. For a Wide Receiver, it is important to consider how strong the pass blocking is, and how strong the Quarterback is.

It is also important to consider a player's role on the team. Is the player a starter or a backup? Related to this, it is important to consider the strength of one's teammates. For a given Running Back, if a teammate is better at running the ball, this may take away from how much the player sees the field. For a given Wide Receiver, if a teammate is better at catching the ball, this may take some targets away from the player. However, the team's top defensive back is often matched up against the team's top Wide Receiver. So, if the team's top Wide Receiver is matched up against a particularly strong Defensive Back, the second- and third-best Wide Receivers may more targets than usual.

It is also important to consider a player's opportunity and usage, which are influenced by many factors, including the skill of the player, the skill of their teammates, the role of the player on the team, the coaching style, the strategy of the opposing team, game scripts, etc. In terms of the player's opportunity and usage, how many snaps do they get? How many touches and/or targets do they receive? Being on the field for more snaps and receiving more touches and/or targets means that the player has more opportunities to score fantasy points. Are they targeted in the red zone? Red zone targets are more likely to lead to touchdown scoring opportunities, which are particularly valuable in fantasy football.

6.10 Matchups

Matchup-related factors to consider include:

- strength of schedule
- weekly matchup

Another aspect to consider is how challenging their matchup(s) and strength of schedule is. For a Quarterback, it is valuable to consider how strong the oppenent's passing defense is. For a Running Back, how strong is the running defense? For a Wide Receiver, how strong is the passing defense and the Defensive Back that is likely to be assigned to guard them?

6.11 Cognitive and Motivational Factors

Other factors to consider include cognitive and motivational factors. Some coaches refer to these as the "X Factor" or "the intangibles." However, just as any other construct in psychology, we can devise ways to operationalize them. Insofar as they are observable, they are measurable.

Cognitive and motivational factors one could consider include:

- reaction time
- knowledge and intelligence
- work ethic and mental toughness
- incentives
  - contract performance incentives
  - whether they are in a contract year

A player's knowledge, intelligence, and reaction time can help them gain an upper-hand even when they may not be the fastest or strongest. A player's work ethic and mental toughness may help them be resilient and persevere in the face of challenges. Contact-related incentives may lead a player to put forth greater effort. For instance, a contract may have a performance incentive that provides a player greater compensation if they achieve a particular performance milestone (e.g., receiving yards). Another potential incentive is if a player is in what is called their "contract year" (i.e., the last year of their current contract). If a player is in the last year of their current contract, they have an incentive to perform well so they can get re-signed to a new contract.

6.12 Fantasy Value

6.12.1 Sources From Which to Evaluate Fantasy Value

There are several sources that one can draw upon to evaluate a player's fantasy value:

- expert or aggregated rankings
- layperson rankings
  - players' Average Draft Position (ADP) in other league snake drafts
  - players' Average Auction Value (AAV) in other league auction drafts
- expert or aggregated projections

6.12.1.1 Expert Fantasy Rankings

Fantasy rankings (by so-called "experts") are provided by many sources. To reduce some of the bias due to a given source, some services aggregate projections across sources, consistent with a "wisdom of the crowd" approach. FantasyPros[1] aggregates fantasy rankings across sources. Fantasy Football Analytics[2] creates fantasy rankings from projections that are aggregated across sources (see the webapp here: https://apps.fantasyfootballanalytics.net).

6.12.1.2 Layperson Fantasy Rankings: ADP and AAV

Average Draft Position (ADP) and Average Auction Value (AAV), are based on league drafts, mostly composed of everyday people. ADP is based on snake drafts, whereas AAV is based on auction drafts. Thus, ADP and AAV are consistent with a "wisdom of the crowd" approach, and I refer to them as forms of rankings by laypeople. ADP data are provided by FantasyPros[3]. AAV data are also provided by FantasyPros[4].

6.12.1.3 Projections

Projections are provided by various sources. Projections (and rankings, for that matter) are a bit of a black box. It is often unclear how they were derived by a particular source. That is, it is unclear how much of the projection was based on statistical analysis versus conjecture.

To reduce some of the bias due to a given source, some services aggregate projections across sources, consistent with a "wisdom of the crowd" approach. Projections that are aggregated across sources are provided by Fantasy Football Analytics[5] (see the webapp here: https://apps.fantasyfootballanalytics.net) and by FantasyPros[6]. Moreover, sites also provide season-long player futures for the over/under betting lines, which are based on Vegas lines and how people bet: https://www.rotowire.com/betting/nfl/player-futures.php.

6.12.1.4 Benefits of Using Projections Rather than Rankings

It is important to keep in mind that rankings, ADP, and AAV are specific to roster and scoring settings of a particular league. For instance, in point-per-reception (PPR) leagues, players who catch lots of passes (Wide Receivers,

[1] https://www.fantasypros.com/nfl/rankings/consensus-cheatsheets.php
[2] https://fantasyfootballanalytics.net
[3] https://www.fantasypros.com/nfl/adp/overall.php
[4] https://www.fantasypros.com/nfl/auction-values/calculator.php
[5] https://fantasyfootballanalytics.net
[6] https://www.fantasypros.com/nfl/auction-values/calculator.php

Tight Ends, and some Running Backs) are valued more highly. As another example, Quarterbacks are valued more highly in 2-Quarterback leagues. Thus, if using rankings, ADP, or AAV, it is important to find ones from leagues that mirror—as closely as possible—your league settings.

Projected statistics (e.g., projected passing touchdowns) are agnostic to league settings and can thus be used to generate league-specific fantasy projections and rankings. Thus, projected statisitics may be more useful than rankings because they can be used to generate rankings for your particular league settings. For instance, if you know how many touchdowns, yards, and interceptions a Quarterback is a projected to throw (in addition to any other relevant categories for the player, e.g., rushing yards and touchdowns), you can calculate how many fantasy points the Quarterback is expected to gain in your league (or in any league). Thus, you can calculate ranking from projections, but you cannot reverse engineer projections from rankings.

6.12.2 Indices to Evaluate Fantasy Value

Based on the sources above (rankings, ADP, AAV, and projections), we can derive multiple indices to evaluate fantasy value. There are many potential indices that can be worthwhile to consider, including a player's:

- dropoff
- value over replacement player (VORP)
- uncertainty

6.12.2.1 Dropoff

A player's *dropoff* is the difference between (a) the player's projected points and (b) the projected points of the next-best player at that position.

6.12.2.2 Value Over Replacement Player

Because players from some positions (e.g., Quarterbacks) tend to score more points than players from other positions (e.g., Wide Receivers), it would be inadvisable to compare players across different positions based on projected points. In order to more fairly compare players across positions, we can consider a player's value over a typical replacement player at that position (shortened to "value over replacement player"). A player's *value over a replacement player* (VORP) is the difference between (a) a player's projected fantasy points and (b) the fantasy points that you would be expected to get from a typical bench player at that position. Thus, VORP provides an index of how much added value a player provides.

6.12.2.3 Uncertainty

A player's *uncertainty* is how much variability there is in projections or rankings for a given player across sources. For instance, consider a scenario where three experts provide ratings about two players, Player A and Player B. Player A is projected to score 300, 310, and 290 points by experts 1, 2, and 3, respectively. Player B is projected to score 400, 300, and 200 points by experts 1, 2, and 3, respectively. In this case, both players are (on average) projected to score the same number of points (300).

```
exampleData <- data.frame(
  player = c(rep("A", 3), rep("B", 3)),
  expert = c(1:3, 1:3),
  projectedPoints = c(300, 310, 290, 400, 300, 200)
)

playerA_mean <- mean(exampleData$projectedPoints[which(exampleData$player == "A")])
playerB_mean <- mean(exampleData$projectedPoints[which(exampleData$player == "B")])

playerA_sd <- sd(exampleData$projectedPoints[which(exampleData$player == "A")])
playerB_sd <- sd(exampleData$projectedPoints[which(exampleData$player == "B")])

playerA_cv <- playerA_mean / playerA_sd
playerB_cv <- playerB_mean / playerB_sd
```

```
playerA_mean
```

```
[1] 300
```

```
playerB_mean
```

```
[1] 300
```

However, the players differ considerably in their uncertainty (i.e., the source-to-source variability in their projections), as operationalized with the standard deviation and coefficient variation of projected points across sources for a given player.

```
playerA_sd
```

```
[1] 10
```

```
playerB_sd
```

```
[1] 100
```

```
playerA_cv
```

```
[1] 30
```

```
playerB_cv
```

```
[1] 3
```

Here is a depiction of a density plot of projected points for a player with a low, medium, and high uncertainty:

```
playerA <- rnorm(1000000, mean = 150, sd = 5)
playerB <- rnorm(1000000, mean = 150, sd = 15)
playerC <- rnorm(1000000, mean = 150, sd = 30)

mydata <- data.frame(playerA, playerB, playerC)

mydata_long <- mydata %>%
  pivot_longer(
    cols = everything(),
    names_to = "player",
    values_to = "points"
  ) %>%
  mutate(
    name = case_match(
      player,
      "playerA" ~ "Player A",
      "playerB" ~ "Player B",
      "playerC" ~ "Player C",
    )
  )

ggplot2::ggplot(
  data = mydata_long,
  ggplot2::aes(
    x = points,
    fill = name
  )
) +
```

```
ggplot2::geom_density(alpha = .3) +
ggplot2::labs(
  x = "Players' Projected Points",
  title = "Density Plot of Projected Points for Three Players"
) +
ggplot2::theme_classic() +
ggplot2::theme(legend.title = element_blank())
```

Figure 6.1 Density Plot of Projected Points for Three Players

Uncertainty is not necessarily a bad characteristic of a player's projected points. It just means we have less confidence about how the player may be expected to perform. Thus, players with greater uncertainty are risky and tend to have a higher upside (or ceiling) and a lower downside (or floor).

6.13 Signs Versus Samples

When thinking about the most important domains to assess for evaluating players and predicting their performance, it is important to distinguish between signs and samples (Den Hartigh et al., 2018). *Signs* are indicators of

underlying states. The signs approach to assessment involves assessing processes that may predict performance, where the emphasis is on what the sign indicates about the underlying attribute, rather than on the specific behavior itself. In football, the signs approach might involve measuring skills using separate tests where players' skills are tested in isolation. For instance, the NFL Combine takes a signs approach to assessment, in which players have a separate test to assess for speed (e.g., 40-yard dash) and jumping ability (e.g., vertical jump).

Samples reflect behaviors that are close to the behavior of interest. The samples approach to asesssment tries to assess the person's performance in a behavior, situation, and context that are as reflective as possible of the kinds of behaviors, situations, and contexts that the person would have to perform in for the position or occupation. In the samples approach, the behavior itself is the main emphasis, rather than an underlying attribute. In football, the samples approach might involve measuring the player's performance during games or game-like situations. For instance, to assess the skills of a Wide Receiver, you might observe—during games or game-like situations—how well they are able to catch passes when closely guarded or double-teamed, or how well they are able to catch poorly thrown passes.

In general, samples tend to be stronger than signs for predicting player performance in sports (Den Hartigh et al., 2018). For instance, compared to tests of speed, power, and agility at the NFL Combine, collegiate performance is a stronger predictor of performance in the NFL (Lyons et al., 2011). That is, previous sports performance is the best predictor of future performance (for a review, see Den Hartigh et al., 2018).

According to the theoretical perspective to sports performance known as the ecological dynamics approach, successful performance in sports involves the coordination of multiple, intertwined skills that are contextually embedded (Den Hartigh et al., 2018). For instance, a Wide Receiver needs to coordinate skills in speed, route running, jumping, and good hands to create space from defenders and to catch poorly thrown passes in a game. The ecological dynamics approach considers the important interaction of the player, task, and environment. Thus, to assess players in a way that is likely to be most predictive of their future performance, it is important for the assessment to retain the player–task–environment interaction (Den Hartigh et al., 2018). For example, it would be valuable to use assessments from games or game-like situations in which players must perform tasks that leverage multiple, intertwined skills and that are similar to the tasks you want to predict.

6.14 Putting it Altogether

After performing an evaluation of the relevant domain(s) for a given player, one must integrate the evaluation information across domains to make a judgment about a player's overall value. When considering how much weight to give to each of various factors, it is important to evaluate how much predictive validity each factor has for predicting a player's successful performance in the NFL. Then, one can weight each variable according to its predictive validity using an actuarial approach. Actuarial approaches are described in a later chapter, in Section 15.2.2. For now, suffice it to say that actuarial approaches leverage statistical formulas, as opposed to using judgment alone. As described in Chapter 14, people's judgment—including judgments by experts (e.g., professional scouts and coaches)—tends to be riddled with biases (Den Hartigh et al., 2018). Moreover, experts tend to disagree in terms of their judgments/predictions about players (Den Hartigh et al., 2018). In addition, there are benefits to leveraging multiple perpectives, consistent with the "wisdom of the crowd", as described in Section 22.3. In general, as noted in Section 6.13, give more weight to samples of relevant behavior—such as past performance in games—than to signs such as NFL Combine metrics (e.g., bench press, 40-yard dash, etc.). Where you can, use assessments from games or game-like situations in which players must perform tasks that leverage multiple, intertwined skills and that are similar to the tasks you want to predict.

When thinking about a player's value, it can be worth thinking of a player's upside and a player's downside. Players that are more consistent may show higher downside but a lower upside. Younger, less experienced players may show a higher upside but a lower downside.

The extent to which you prioritize a higher upside versus a higher downside may depend on many factors. For instance, when drafting players, you may prioritize drafting players with the highest downside (i.e., the safest players), whereas you may draft sleepers (i.e., players with higher upside) for your bench. When choosing which players to start in a given week, if you are predicted to beat a team handily, it may make sense to start the players with the highest downside. By contrast, if you are predicted to lose to a team by a good margin, it may make sense to start the players with the highest upside.

7

The Fantasy Draft

7.1 Getting Started

7.1.1 Load Packages

7.2 Types of Fantasy Drafts

There are several types of drafts in fantasy football. The most common types of drafts are snake drafts and auction drafts.

7.2.1 Snake Draft

In a snake draft, the participants (i.e., managers) are assigned a draft order. In the first round, the managers draft in that order. In the second round, the managers draft in reverse order. It continues to "snake" in this way, round after round, so that the person who has the first pick in a given round has the last pick in the next round, and whoever has the last pick in a given round has the first pick in the next round.

7.2.2 Auction Draft

In an auction draft, the managers are assigned a nomination order and there is a salary cap (e.g., $200). The first manager chooses which player to nominate. Then, the managers bid on that player like in an auction. In order to bid, the manager must raise the price by at least $1. If two managers want to obtain the same player, they may continue to raise the amount until one manager backs out and is no longer to bid by raising the price. The highest bidder wins (i.e., drafts) that player. Then, the second manager nominates a player, and the managers bid on that player. This process repeats until all teams have drafted their allotment of players.

7.2.3 Comparison

Snake drafts are more common than auction drafts. Snake drafts tend to be quicker than auction drafts. However, auction drafts are more fair than snake drafts. In an auction draft, unlike a snake draft, all players are available to all teams. For instance, in a snake draft, the first 9 players drafted are unavailable to the 10th pick of the first round. So, if you have the 10th pick and want the top-ranked player, this player would not be available to you in the snake draft. However, in the auction draft, every player is available to every manager, so long as the manager is able and willing to bid enough.

7.3 Draft Strategy

7.3.1 Overview

There is no one "right" draft strategy. As noted by Lee & Liu (2022) in their analysis of fantasy drafts, the effectiveness of any draft strategy depends on the strategies of the other managers in the league. Sometimes it works best to "zig" when everyone else is "zagging". For instance, Lee & Liu (2022) found that the most successful draft strategies in terms of roster composition (i.e., the number of players for each position) were not the most common draft strategies. For instance, if you notice that everyone else is drafting Wide Receivers, this may mean that other managers are over-valuing Wide Receivers, and this could be a nice opportunity to draft a Running Back for good value.

In general, you will first want to generate the rankings you will use to select which players to prioritize. You may generate your rankings based one or more of the following:

- your evaluation of players[1]
- expert or aggregated rankings
- layperson rankings
  - players' Average Draft Position (ADP) in other league drafts (for snake drafts)
  - players' Average Auction Value (AAV) in other league drafts (for auction drafts[2])
- expert or aggregated projections
- indices derived from rankings and projections

[1] player-evaluation.qmd
[2] sec-draftStrategyAuction

As described in Section 22.3, there can be benefits of leveraging the wisdom of the crowd by using rankings or projections that are averaged across many people and perspectives. Section 6.12.1 describes where to obtain aggregated rankings, aggregated projections, ADP, and AAV data. As described in Chapter 15, there are also benefits to using the actuarial approach to prediction rather than (merely) using judgment.

It is not sufficient to compare players in terms of projected points because some positions have more depth than other positions. Some positions show *positional scarcity*—that is, a limited number of high performing players.

An important concept in the draft is "dropoff", which is described in Section 6.12.2.1. *Dropoff* at a given position, is the difference—in terms of projected fantasy points—between (a) the best available player remaining at that position and (b) the second-best available player remaining at that position. If there is a bigger dropoff at a given position, there may be greater value in drafting the top player from that position. For instance, consider the following scenario: "Quarterback A" is projected to score 325 points, and "Quarterback B" is projected to score 320 points. "Tight End A" is projected to score 230 points, and "Tight End B" is projected to score 150 points. In this example, there is a much greater dropoff for Tight Ends than there is for Quarterbacks. Thus, even though "Quarterback A" is projected to score more points than "Tight End A", "Tight End A" may be more valuable, relatively, because there is still a good Quarterback available if someone else drafts "Quarterback A". In general, Kickers and Defenses tend to have the lowest dropoff (i.e., the lowest expected drop in fantasy points) by positional rank, so it makes sense to draft Kickers and Defenses late in the draft (Lee & Liu, 2022). Defenses, in particular, appear to be among the least predictable of the positions (Lee & Liu, 2022).

Another important concept is a player's value over a typical replacement player at that position (shortened to "value over replacement player"; VORP), which is described in Section 6.12.2.2. A player's value over a typical replacement player provides a way to more fairly compare (and thus rank) players across different positions.

Another important concept is a player's uncertainty, which is described in Section 6.12.2.3.

In both snake and auction draft formats, your goal is to draft the team whose weekly starting lineup scores the most points and thus the collection of players with the greatest VORP. For your starting lineup, it may make sense—especially with your earliest selections—when comparing two players with equivalent VORP, to prioritize players with higher consistency and lower uncertainty, because they may be considered "safer" with a higher floor. However, when drafting players for your bench, it make make more sense to prioritize high-risk, high reward players with greater uncertainty, because they may have a higher ceiling. Players with a higher ceiling have a potential to be

"sleepers"—players who are valued low (i.e., with a high ADP or low AAV) and who outperform their valuation. Note that, although players with greater uncertainty are high-risk, high-reward players, selecting this kind of a player for your bench (i.e., in a late round or for a small cost) is a *lower* risk selection, because you have less to lose with later/lower-cost picks. That is, even though the *player* is higher risk, selecting a higher risk player for your bench is a lower risk *decision*.

The Spurs in the National Basketball Association (NBA) were well-reputed for excelling in this draft strategy[3] (archived at https://perma.cc/X7NW-WZC6). They frequently used their second-round picks to draft high-risk, high-reward players. Sometimes, the secound round pick was a bust, but they have little to lose with a failed second round pick. Other times, their second round picks—including Willie Anderson, DeJuan Blair, Goran Dragic, Luis Scola, and Manu Ginóbili—greatly outperformed expectations. Thanks, in part, to this draft strategy, the team showed strong extended success for nearly three decades from 1989 through the late-2010s.

However, the draft strategies to achieve the "optimal lineup" differ between snake versus auction drafts.

One factor that is *not* included above is whether a player is on your favorite team. Managers commonly like to draft players on their favorite teams (e.g., Cowboys, Eagles). That is fine—fantasy football is a game. Do what is fun for you. However, if your goal is to select the best players, leave your allegiances at the door. Selecting players based on their playing for your favorite team—rather than based on performance—is a form of cognitive bias.

In general, the most consequential decisions tend to be those made early in the draft regarding the top projected players (Lee & Liu, 2022). That is, the earlier selections tend to have the greatest impact on the success of one's fantasy season. So, make sure to spend time in your draft preparation to identify the players you want to select early in the draft. Moreover, some teams like to hedge their bet on their top Running Backs in the case that the player were to get injured, by drafting the player's backup, a strategy known as *handcuffing*. However, there is not strong evidence that handcuffing leads to better outcomes (Lee & Liu, 2022).

In general, Lee & Liu (2022) found that teams that drafted more Running Backs and Wide Receivers tended to outperform other teams.

[3] https://harvardsportsanalysis.org/2013/11/beating-the-nba-draft-does-any-team-outperform-expectations/

7.3.2 Snake Draft

In general, your goal is to draft the team whose weekly starting lineup has the greatest VORP. Consequently, you are often looking to pick the player with the highest VORP at a given selection, while keeping in mind (a) the dropoff of players at other positions and (b) which players may be available at subsequent picks so that you do not sacrifice too much later value with a given selection. For instance, if a particular Quarterback has a slightly higher VORP than a particular Running Back, but the Quarterback is likely to be available at the manager's next pick but the Running Back is likely to be unavailable at their next pick, it might make more sense to draft the Running Back.

Herding behavior is common in snake drafts. As described in Section 22.3, herding occurs when people align their behavior with others. For instance, Lee & Liu (2022) found evidence that when Quarterbacks, Kickers, and Defenses were drafted by one team, subsequent teams were more likely to draft a player of that position. The same was not the case for Running Backs, Wide Receivers, and Tight Ends.

7.3.3 Auction Draft

According to an analysis[4] by the Harvard Sports Analysis Collective (archived at https://perma.cc/P7RX-92UU), the majority of the manager's salary cap should be spent on the starting lineup, and you should spend less on bench players. This is known as the "stars and scrubs" draft strategy. Based on the analysis, the author recommended applying a 10% premium to the top players and a 10% discount to the lower-tiered players. The idea behind the approach is that a player on your bench does not contribute to the team's points and, thus, most players drafted to your bench do not contribute much to the team's points throughout the season. That said, bench players can be important in the case of a starter's injury or under-performance. So, it is recommended to draft starters with lower uncertainty who are safer. In contrast to your starting lineup, you may look to draft players on your bench who have greater uncertainty for their high reward potential in a low-risk selection given the lower price.

An alternative to the "stars and scrubs" approach is to wait to draft more "high-value" players after other managers have over-paid for players. In any case, having some small amount of cap left over toward the end of the draft can help you draft good value players to fill out your bench spots for cheap (e.g., $2). Having some depth can help you offset the risk of injuries and the

[4] https://harvardsportsanalysis.wordpress.com/wp-content/uploads/2012/04/fantasyfootballdraftanalysis1.pdf

possibility that some of your starters may underperform their expectation, which is quite likely given how challenging it is to predict fantasy performance.

8

Research Methods

8.1 Getting Started

8.1.1 Load Packages

8.2 Sample vs Population

In research, it is important to distinguish between the sample and the target population. The target *population* is who you want your study's findings to generalize to. For instance, if we want our findings to lead to inferences we can draw regarding all current NFL players, then NFL players are our target population. However, despite our best efforts to recruit all NFL players into our study, we may not succeed in doing that. The participants (i.e., people or players) who we successfully recruit to be in our study represent our *sample*. The number of participants in the study is our *sample size*.

It is rare for the sample to include all people who are in the target population. It can be costly to recruit large samples, and many potential participants may decline to participate for a variety of reasons (insufficient time, lack of interest in the study, distrust of scientists, etc.). Thus, our goals are (a) to recruit as many people from the population as possible and (b) for the sample to be as *representative* of the population as possible.

For increasing the representativeness of the sample (with respect to the population), we might conduct a *random sample*, in which each person in the population (i.e., each NFL player) has equal likelihood of being selected. For instance, we might randomly select 250 players to recruit to the study. True random samples, though strong in aspiration, are difficult and costly to achieve. In reality, many researchers conduct convenience sampling. A convenience sample is recruited because it is convenient (i.e., less costly and time-consuming).

For instance, many studies examine college students—in part, because they are easy to recruit. If our target population is NFL players but we are unable

to recruit NFL players into our study, we could easily recruit a large sample of college students. Although the convenience sample may afford a very large sample, the college student sample may not be representative of the target population (NFL players). Thus, the findings in our study may not *generalize* to NFL players—that is, what we learn in college students may not apply in the same way among NFL players. For instance, if we learn that consumption of sports drinks (compared to drinking only water) improves running speed among college students, that may not be the case among NFL players.

8.3 Research Designs

There are three broad types of research designs:

- experiment
- correlational/observational study
- case study

8.3.1 Experiment

In an *experiment*, there are one or more things (i.e., variables) that we manipulate to see how the manipulation influences the process of interest. The variable that we manipulate is the *independent variable*. By contrast, the *dependent variable* is the variable that we evaluate to determine whether it was influenced by the manipulation (i.e., by the independent variable).Besides the independent and dependent variables, the researcher attempts to hold everything else constant through processes including standardization and random assignment. *Standardization* involves using the same procedures to assess each participant, so that scores can be fairly compared across participants (and groups). Random assignment involves randomly assigning participants to conditions of the independent variable, so the people in each condition are comparable and do not differ systematically.

8.3.1.1 Intervention Study

An intervention study is a study that involves some modification (e.g., a treatment) with the intent to improve people's standing on the dependent variable (e.g., depression). Some intervention studies have a control group, whereas intervention studies do not. Inclusion of a control group is valuable; without a control group, you do not know whether any apparent gains in the treatment

condition were due to the treatment per se versus just the mere passage of time, regression effects, or other things that were going on in the participants' lives. An intervention that includes random assignment (e.g., to the intervention or control group) is an experiment. A randomized controlled trial (RCT) is an example of an experiment because it is an intervention with random assignment.

For instance, we may be interested to evaluate whether players perform better (e.g., run faster) if they drink a sports drink compared when they drink only water. Our hypothesis might be that players will be expected to perform better when they drink a sports drink (compared to when they drink only water). To this this research question and hypothesis, we might conduct an experiment by randomly assigning some players during practice to receive a sports drink and some players to receive only water. In this case, our independent variable is whether the player receives a sports drink. Our dependent variable might be their 40-yard dash time during practice.

8.3.2 Correlational/Observational Study

In a correlational (aka observational) study, we do not manipulate a variable to see how the manipulation influences another variable. Instead, we examine how two variables, a predictor and an outcome variable, are associated. The hypothesized cause is called the predictor variable. The hypothesized effect is called the outcome variable. In this way, the predictor variable is similar to the independent variable, and the outcome variable is similar to the dependent variable. However, unlike the independent and dependent variables in an experiment, the predictor and outcome variables in a correlational study are not manipulated.

For instance, to use a correlational study to test the possibility that players who drink sports drinks perform better than players who drink only water, we could examine whether the players who drink sports drinks during a game score more fantasy points than players who drink only water during the game. In this case, our predictor variable is whether the players drinks sports drinks during a game. Our outcome variable is the number of fantasy points the player scored.

8.3.2.1 Correlation Does Not Imply Causation

As the maxim goes, "correlation does not imply causation"—just because two variables are associated does not necessarily mean that they are causally related.

Just because `X` is associated with `Y` does not mean that `X` causes `Y`. Consider that you find an association between variables `X` and `Y`.

There are several reasons why you might observe an association between X and Y:

- X causes Y
- Y causes X
- X and Y are bidirectional: X causes Y and Y causes X
- a third variable (i.e., confound), Z, influences both X and Y
- the association between X and Y is spurious

For instance, one possibility is that the association we observed reflects our hypothesis that X causes Y, as depicted in Figure 8.1. That is, consumption of more sports drink may improve players' performance.

Figure 8.1 Hypothesized Causal Effect Based on an Observed Association Between X and Y, Such That X Causes Y.

However, a second possibility is that the association reflects the opposite direction of effect, where Y actually causes X, as depicted in Figure 8.2. For instance, greater performance may lead players to drink more sports drink (rather than the reverse).

Figure 8.2 Reverse (Opposite) Direction of Effect From the Hypothesized Effect, Where Y Causes X.

A third possibility is that the association reflects a bidirectional effect, where X causes Y and Y causes X, as depicted in Figure 8.3. For instance, consumption of more sports drink may improve players' performance, and greater performance in turn may lead players to drink more sports drink.

A fourth possibility is that the association could reflect the influence of a third variable. If a third variable is a common cause of each and accounts for their association, it is a *confound*. An observed association between X and Y

Figure 8.3 Bidirectional Effect Between `X` and `Y`, such that `X` causes `Y` and `Y` causes `X`.

could reflect a confound—i.e., a cause (`Z`) that influences both `X` and `Y`, which explains why `X` and `Y` are correlated even though they are not causally related. A third variable confound that is a common cause of both `X` and `Y` is depicted in Figure 8.4. For instance, it may not be that sport drink consumption per se influences player performance; rather, it may be that players who are more intelligent or have more financial resources tend to drink more sports drinks and also tend to perform better. In this case, intelligence or financial resources may be a confound that influences both sports drink consumption and player performance, but sports drink consumptions—though correlated with player performance—does not influence player performance.

For another example, consider that ice cream sales are associated with shark attacks. It is unlikely that more people eating ice creams leads to shark attacks. There is a likely a third variable—heat waves—that is a confound because it influences both ice cream sales and shark attacks and explains their association.

Lastly, the association might be spurious. It might just reflect random variation (i.e., chance), and that when tested on an independent sample, what appeared as an association in the original dataset may not hold when testing the association in a new dataset.

8.3.3 Case Study

In a case study, we assess a small sample of individuals (commonly only one person or a few people), often with rich qualitative information. Themes may be coded from the qualitative information, which may help inform inferences about whether some process may have played a role in influencing the outcome of interest. The inferences are then drawn in a subjective, qualitative way. Testimonials and anecdotes are examples are case studies.

For instance, to use a case study to evalute the possibilty that players who drink sports drinks perform better than players who drink only water, we could conduct an in-depth interview with a player. In the interview, we might ask the player how they performed in games with versus without a sports drink

Figure 8.4 Confounded Association Between X and Y due to a Common Cause, Z.

and have them discuss whether they believe the sports drink improved their performance (and if so, how). Then, based on the player's responses, we might code the responses to extract themes and to make a qualitative judgement of whether or not the player likely performed better during games in which they had a sports drink.

8.3.4 Other Features of the Research Design

8.3.4.1 Number of Timepoints

In addition to whether the research design is an experiment, correlational/observational study, or a case study, a research design can also have one or multiple timepoints. The differing number of timepoints allow studies to be characterized as one of the following:

- cross-sectional
- longitudinal

8.3.4.1.1 Cross-Sectional

A *cross-sectional study* is a study with one timepoint.

For instance, in a cross-sectional study evaluating whether having a sports drink improves player performance, we might assess players' drinking behavior and performance during only game 1.

Cross-sectional studies are more common than longitudinal studies because cross-sectional studies are less costly and time-consuming. They can provide a helpful starting point to test findings more rigorously in subsequent longitudinal studies.

8.3.4.1.2 Longitudinal Design

A *longitudinal study* is a study with more than one timepoint. When the same measures are assessed at each of multiple timepoints, we refer to this as a "repeated measures" design.

In a longitudinal study evaluating whether having a sports drink improves player performance, we might assess players' drinking behavior and performance during each game of the season, and possibly across multiple seasons.

Longitudinal studies are less common than cross-sectional studies because longitudinal studies are more costly and time-consuming. Nevertheless, longitudinal studies can allow us test our hypotheses more rigorously, because they can allow us to test whether changes in the predictor/indepdnent variable leads to changes in the outcome/dependent variable. Thus, compared to cross-sectional studies, longitudinal studies can provide greater confidence in causal inferences.

8.3.4.2 Within- or Between-Subject

A research design can also be within-subject, between-subject, or both. A study can involve both within-subject and between-subject comparisons if one predictor/independent variable is within-subject and another predictor/independent variable is between-subject.

8.3.4.2.1 Within-Subject Design

A *within-subject design* is one in which each participant (i.e., person or player) receives multiple levels of the independent variable (or predictor).

For instance, in an experiment evaluating whether having a sports drink improves player performance, we might assign players to drink the sports drink

in the first half of the game and to drink only water in the second half of the game. Or we could assign some of the players to drink sports drink in the first half and water in the second half, and assign the other players to drink water in the first half and sports drink in the second half.

In a correlational study evaluating whether having a sports drink improves player performance, we might evaluate how within-person changes in sports drink consumption are associated with within-person changes in performance. That is, we could evaluate, when a given player has a sports drink (or more sports drinks), do they perform better than when the same individual has only water (or fewer sports drinks)?

Within-subject designs tend to have greater statistical power than between-subject designs. However, within-subject designs often have *carryover effects*. For instance, consider the study in which we assign players to drink only water in the first and third quarters and to drink sports drink in the second and fourth quarters (an A-B-A-B design). Drinking sports drink in the second quarter could increase how much hydration a player has throughout the rest of the game, which could lead to altered performance in the third and fourth quarters that is not due to what they drink in third and fourth quarters.

8.3.4.2.2 Between-Subject Design

A *between-subject design* is one in which each participant (i.e., person or player) receives only one level of the independent variable.

For instance, in an experiment evaluating whether having a sports drink improves player performance, we might assign some players to drink the sports drink but the other players to drink only water.

In a correlational study evaluating whether having a sports drink improves player performance, we might evaluate whether people who drink sports drinks tend to perform better than players who drink only water. Or, we could evaluate whether players who drink more sports drinks perform better than players who drink fewer sports drinks (i.e., whether the number of sports drinks consumed during a game is correlated with player performance).

8.4 Research Design Validity

Research design validity involves the accuracy of inferences from a study. There are three types of research design validity:

- internal validity
- external validity
- conclusion validity

8.4.1 Internal Validity

Internal validity is the extent to which we can be confident that the associations identified in the study are causal.

8.4.2 External Validity

External validity is the extent to which we can be confident that findings from the study play out similarly in the real world—that is, the findings generalize to the target population.

8.4.3 Tradeoffs Between Internal and External Validity

There is a tradeoff between internal and external validity—a single research design cannot have both high internal and high external validity. Each study and design has weaknesses. Some research designs are better suited for making causal inferences, whereas other designs tend to be better suited for making inferences that generalize to the real world. The research design that is best suited to making causal inferences is an experiment because it is the design in which the researcher has the greatest control over the variables. Thus, experiments tend to have higher internal validity than other research designs. However, by manipulating one variable and holding everything else constant, the research takes place in a very standardized fashion that can become like studying a process in a vacuum. So, even if a process is theoretically causal in a vacuum, it may act differently in the real world when it interacts with other processes.

Correlational designs have greater capacity for external validity than experimental designs because the participants can be observed in their natural environments to evaluate how variables are related in the real world. However, the greater external validity comes at a cost of lower internal validity. Correlational designs are not well-positioned to make causal inferences. Correlational studies can account for potential confounds using *covariates* or for the reverse direction of effect using longitudinal designs, but the researcher has less control over the variables than in an experiment.

As the internal validity of a study's design increases, its external validity tends to decrease. The greater control we have over variables (and, therefore, have greater confidence about causal inferences), the lower the likelihood that the

findings reflect what happens in the real world because it is studying things in a metaphorical vacuum. Because no single research design can have both high internal and external validity, scientific inquiry needs a combination of many different research designs so we can be more confident in our inferences—experimental designs for making causal inferences and correlational designs for making inferences that are more likely to reflect the real world.

Case studies, because they have smaller sample sizes and inferences drawn in a subjective, qualitative way, tend to have lower external validity than both experimental and correlational studies. Case studies also tend to have lower internal validity because they have less control over variables, and thus fail to remove the possibility of illusory correlations, potential confounds, or the reverse direction of effect. Thus, case studies are among the weakest forms of evidence. Nevertheless, case studies can still be useful for generating hypotheses that can then be tested empirically with a larger sample in experimental or correlational studies.

8.4.4 Conclusion Validity

Conclusion validity is the extent to which a study's conclusions are reasonable about the association among variables based on the data. That is, were the correct statistical analyses performed, and are the interpretations of the findings from those analyses correct?

8.5 Mediation vs Moderation

Both types of effects involve (at least) three variables:

1. An independent/predictor variable, which will be labeled as `X`.
2. A dependent/outcome variable, which will be labeled as `Y`.
3. The mediator or moderator variable, which will be labeled as `M`.

A mnemonic to help remember the difference between mediation and moderation is in Figure 8.5.

Figure 8.5 Mediation Versus Moderation Mnemonic.

8.5.1 Mediation

8.5.1.1 Overview

Mediation is a causal chain of events, where one variable (a mediator variable) at least partially explains (or accounts for) the association between two other variables (the predictor variable and the outcome variable). In mediation, a predictor (`X`) leads to a mediator (`M`), which leads to an outcome (`Y`). Mediation answers the question of, "**Why (or how)** does `X` influence `Y`? A mediator (`M`) is a variable that helps explain the assocation between two other variables, and it answers the question of why/how `X` influences `Y`. That is, the mediator is the variable that helps explain how/why `X` is related to `Y`. In other words, you can think of the mediator as the mechanism that helps explain why `X` has an impact on `Y`. The association between `X` and `Y` gets smaller when accounting for `M`. Visually this can be written as in Figure 8.6:

where `X` is causing `M`, which in turn is causing `Y`. In other words, `X` leads to `M`, and `M` leads to `Y`.

For instance, if we determine that consuming sports drinks improves player

Figure 8.6 Mediation.

performance, we may want to know how/why. That is, what is the mechanism that leads consumption of sports drinks to improve player performance? We might hypothesize that consumption of sports drink helps increase a player's hydration, which in turn will improve the player's performance. In this case, increased hydration mediates (i.e., helps explain or account for) the effect of the sports drink consumption on improved player performance.

Question: Why/how does sports drink consumpion lead players to perform better?
Answer: increased hydration

As a picture, we can draw this assocation as in Figure 8.7:

Figure 8.7 Mediation Example.

8.5.1.2 Types of Mediation

8.5.1.2.1 Full Mediation

When one mechanism fully accounts for the effect of the predictor variable on the outcome variable, this is known as **full mediation**, as depicted in Figure 13.15:

Figure 8.8 Full Mediation.

8.5.1.2.2 Partial Mediation

When a single process partially—but does not fully—accounts for the effect of the predictor variable on the outcome variable; this is known as **partial mediation** and is depicted in Figure 13.16:

Figure 8.9 Partial Mediation.

8.5.1.2.3 Multiple Mediators

In addition, there can be multiple mediators/mechanisms that account for the effect of a predictor variable on an outcome variable, as depicted in Figure 8.10:

8.5.2 Moderation (i.e., Interaction)

8.5.2.1 Overview

Moderation (sometimes called an "interaction"), on the other hand, occurs when there is a variable or condition (`M`; called a "moderator") that changes the assocation between `X` and `Y`. That is, the effect of the predictor variable on the outcome variable differs at different levels of the moderator variable. In these cases, `X` and `M` work together to have an effect on `Y`; here `X` does not have a direct effect on `M`. Moderation answers the question of, "**For whom** does `X` influence `Y`?" If `X` influences `Y` more strongly for some people or in some circumstances, we would say that there is an interaction such that the effect of `X` on `Y` depends on `M`, as depicted in Figure 8.11:

For example, if the effect of consuming sports drinks on player performance

Figure 8.10 Multiple Mediators.

Figure 8.11 Moderation.

differs for Quarterbacks and Wide Receivers, the interaction could be depicted in Figures 8.12 and 8.13:

An interaction can be identified visually by non-parallel lines at different levels of the moderator. In this example, the player's position moderates the effect consuming sports drinks on player performance. In particular, there is a strong positive association between consuming sports drinks and player performance for Wide Receivers (as evidenced by the upward slope of the best-fit regression line), whereas there is no association between consuming sports drinks and player performance for Quarterbacks (as evidenced by the flat line).

Figure 8.12 Moderation Example: Path Diagram.

Figure 8.13 Moderation Example: Interaction Graph.

8.6 Levels of Measurement

It is important to know the levels of measurement of your data, because the level(s) of measurement of your data constrain the types of comparisons and analyses that you can meaningfully perform. There are four levels of measurement that any variable can have:

- nominal
- ordinal
- interval
- ratio

Each is described below:

8.6.1 Nominal

A variable is considered nominal if it is composed of qualitative classifications. You cannot meaningfully evaluate whether one number in the variable is larger than another number in the variable because higher numbers do not reflect higher levels of the concept. Examples of nominal variables include:

- sex (e.g., 1 = male; 2 = female)
- race (e.g., 1 = American Indian; 2 = Asian; 3 = Black; 4 = Pacific Islander; 5 = White)
- ethnicity (e.g., 0 = Non-Hispanic/Latino; 1 = Hispanic/Latino)
- zip code
- jersey number

A football player's jersey number is an example of a nominal variable. A jersey number of 7 is not higher on whatever concept of interest compared to a jersey number of 6.

To examine the central tendency of a nominal variable, you can determine the mode, but you cannot calculate a mean or median.

8.6.2 Ordinal

A variable is considered ordinal if the classifications are ordered. However, ordinal variables do not have equally spaced intervals. Examples of ordinal intervals include:

- likert response scales (e.g., 1 = strongly disagree; 2 = disagree; 3 = neutral; 4 = agree; 5 = strongly agree)
- educational attainment (e.g., 1 = no formal education; 2 = elementary school; 3 = middle school; 4 = high school; 5 = college; 6 = graduate degree)
- academic grades on A–F scale (e.g., 1 = A; 2 = B; 3 = C; 4 = D; 5 = F)
- player rank (1 = 1st; 2 = 2nd; 3 = 3rd, etc.)

A football player's fantasy rank is an example of an ordinal variable. A player with a fantasy rank of 1 has a higher rank than a player with a rank of 2, but it is not known how far apart each player is—i.e., the intervals do not all reflect the same distance. For instance, the distance between the top-ranked player and the 2nd-best player might be 30 points, whereas the distance between the 2nd-best player and the 3rd-best player might be 2 points.

To examine the central tendency of ordinal data, the median and mode are most appropriate; however, the mean may be used (unlike for nominal data).

8.6.3 Interval

A variable is considered interval if the classifications are ordered (similar to ordinal data) and have equally spaced intervals (unlike ordinal data). However, interval variables do not have a meaningful zero that reflects absence. Examples of interval data include:

- temperature on the Fahrenheit or Celsius scale
- time of day

For instance, the temperature difference between 80 and 90 degrees Fahrenheit is the same as the temperature difference between 90 and 100 degrees Fahrenheit. However, 0 degrees Fahrenheit does not reflect absence of temperature/heat.

Interval data can be meaningfully added or subtracted. For instance, if a game starts at 4 pm and ends at 7 pm, you know the game lasted 3 hours ($7 - 4 = 3$). However, interval data cannot be meaningfully multiplied or divided. For instance, 100 degrees Fahrenheit is not twice as hot as 50 degrees Fahrenheit.

To examine the central tendency of interval data, you can compute the mean, median, or mode.

8.6.4 Ratio

A variable is considered ratio if the classifications are ordered (similar to ordinal data), have equally spaced intervals (like interval data), and have an

absolute zero point that reflects absence of the concept. Examples of ratio data include:

- temperature on the Kelvin scale
- height
- weight
- age
- distance
- speed
- volume
- time elapsed
- income
- stock price
- years of formal education
- points in football

For instance, points in football has order, equally spaced intervals, and an absolute zero—a team cannot score less than zero points, and zero points reflects absence of points (though it could be argued to be interval data because zero points does not reflect absence of skill.)

Ratio data can be meaningfully added, subtracted, multiplied, or divided. A player who weighs 350 pounds weighs twice as much as someone who weighs 175 pounds.

To examine the central tendency of ratio data, you can compute the mean, median, or mode.

8.7 Psychometrics

Below, I provide brief discussions of various aspects of measurement reliability and validity. For more information on these and other aspects of psychometrics, see Petersen (2024b) and Petersen (2024c).

8.7.1 Measurement Reliability

The *reliability* of a measure's scores deals with the *consistency* of measurement. This book focuses on the following types of reliability:

- test–retest reliability

- inter-rater reliability
- intra-rater reliability
- internal consistency
- parallel-forms reliability

For more information on these and other aspects of reliability, see https://isaactpetersen.github.io/Principles-Psychological-Assessment/reliability.html (Petersen, 2024b, 2024c).

8.7.1.1 Test–Retest Reliability

Test–retest reliability evaluates the consistency of scores across time. For a construct that is expected to be stable across time (e.g., hand size in adults), we would expect our measurements to be consistent across time. The consistency of scores across time can be examined in terms of relative or absolute test–retest reliability. Relative test–retest reliability—i.e., the consistency of individual differences across time—is commonly evaluated using the coefficient of stability (i.e., the Pearson correlation coefficent). Absolute test–retest reliability—i.e., the absolute consistency of people's scores across time—is commonly evaluated using the coefficient of repeatability.

8.7.1.2 Inter-Rater Reliability

Inter-rater reliability evaluates the consistency of scores across raters. For instance, if we have a strong measure for assessing college players' aptitude to succeed in the NFL, the measure should yield a similar score for a given player regardless of which (trained) rater (e.g., coach or talent scout) uses it to rate the player. The consistency of scores across raters is commonly evaluated using the intraclass correlation coefficient (for continuous variables) and Cohen's kappa (κ; for categorical variables).

8.7.1.3 Intra-Rater Reliability

Intra-rater reliability evaluates the consistency of scores within a given rater. If we have a strong measure for assessing college players' aptitude to succeed in the NFL, the measure should yield a similar score for a given player from the same (trained) rater (e.g., coach or talent scout) each time they rate the same player (assuming the player's aptitude has not changed). The consistency of scores within raters can be evaluated using similar approaches as those evaluating inter-rater reliability.

8.7.1.4 Internal Consistency

Internal consistency evaluates the consistency of scores across items within a measure. If we develop a strong questionnaire measure to assess a college players' aptitude to succeed in the NFL, the scores should be relatively consistent across items. The consistency of scores across items within a measure is commonly evaluated using Cronbach's alpha (α) or McDonald's omega (ω).

8.7.1.5 Parallel-Forms Reliability

Parallel-forms reliability evaluates the consistency of scores across different but equivalent forms of a measure. If we develop two equivalent versions of the Wonderlic Contemporary Cognitive Ability Test (Form A and Form B) so that players sitting next to each other do not receive the same items, we would expect a player's score on Form A would be similar to their score on Form B. Parallel-forms reliability is is commonly evaluated using the coefficient of equivalence (i.e., the Pearson correlation coefficent).

8.7.2 Measurement Validity

The *validity* of a measure's scores deals with the *accuracy* of measurement. This book focuses on the following types of validity:

- face validity
- content validity
- criterion-related validity
  - concurrent (criterion-related) validity
  - predictive (criterion-related) validity
- construct validity
- convergent validity
- discriminant validity
- incremental validity
- ecological validity

For more information on these and other aspects of validity, see https://isaactpetersen.github.io/Principles-Psychological-Assessment/validity.html (Petersen, 2024b, 2024c).

8.7.2.1 Face Validity

Face validity evaluates the extent to which a measure "looks like" (on its face) it assesses the construct of interest. For instance, if a measure is developed to

assess aptitude of Wide Receivers for the position, it would be considered to have face validity if everyday (lay) people believe that it assesses aptitude for being a successful Wide Receiver.

8.7.2.2 Content Validity

Content validity evaluates the extent to which the measure assesses the full breadth of the content, as determined by context experts. For the measure to have content validity, it should not have gaps (missing content facets) or intrusions (facets of other constructs). For instance, a strong measure for assessing a player's aptitude to succeed in the NFL might need to include a player's speed, strength, size, lateral quickness, etc. If the measure is missing their speed, this would be a content gap. If the measure assesses a construct-irrelevant facet (e.g., their attractiveness), this would be a content intrusion.

8.7.2.3 Criterion-Related Validity

Criterion-related validity evaluates the extent to which the measure's scores are related to meaningful variables of interest. Criterion-related validity is commonly evaluated using a Pearson correlation or some form of regression.

There are two types of criterion-related validity:

- concurrent (criterion-related) validity
- predictive (criterion-related) validity

8.7.2.3.1 Concurrent (Criterion-Related) Validity

Concurrent criterion-related validity (aka concurrent validity) evaluates the extent to which the measure's scores are related to meaningful variables of interest assessed at the same point in time. That is, concurrent validity could evaluate whether current player statistics (e.g., passing yards) are associated with their fantasy points.

8.7.2.3.2 Predictive (Criterion-Related) Validity

Predictive criterion-related validity (aks predictive validity) evaluates the extent to which the measure's scores are related to meaningful variables of interest that are assessed at a later point in time. For example, predictive validity could evaluate whether scores on the measure we developed to assess a player's aptitude to succeed in the NFL predicts later performance in the NFL.

8.7.2.4 Construct Validity

Construct validity evaluates the extent to which the measure's scores accurately assess the construct of interest. If we develop a measure with intent to assess aptitude for being a successful Running Back, and it appears to more accurately assess aptitude for being a successful Wide Receiver, then our measure has poor construct validity for assessing aptitude for being a successful Running Back. Construct validity subsumes convergent and discriminant validity, in addition to all of the other forms of measurement validity.

8.7.2.5 Convergent Validity

Convergent validity evaluates the extent to which the measure's scores are related to other measures of the same construct. For instance, if we develop a new measure to assess intelligence, its scores should be related to scores from other measures designed to assess intelligence (e.g., Wonderlic Contemporary Cognitive Ability Test).

8.7.2.6 Discriminant Validity

Discriminant validity evaluates the extent to which the measure's scores are unrelated to measures of the different constructs. For instance, if we develop a new measure to assess intelligence, its scores should be less strongly associated with measures of other constructs (e.g., measures of happiness).

8.7.2.7 Incremental Validity

Incremental validity evalutes the extent to which the measure's scores provide an increase in predictive accuracy compared to other information that is easily and cheaply available. That is, in order to be useful, a strong measure should tell us something that we did not already know. For instance, if we develop a strong measure of intelligence, it should result in increased predictive accuracy (for success in the NFL) compared to when just relying on the Wonderlic Contemporary Cognitive Ability Test.

8.7.2.8 Ecological Validity

Ecological validity evaluates the extent to which the measures' scores are indicative of the behavior of a person in the natural environment. For instance, measures of a players' speed during a game has higher ecological validity (and is more predictive of their performance) than their speed during the NFL Combine (Lyons et al., 2011). For instance, compared to tests of speed, power, and agility at the NFL Combine, collegiate performance is a stronger

predictor of performance in the NFL (Lyons et al., 2011). That is, previous sports performance is the best predictor of future performance (for a review, see Den Hartigh et al., 2018).

8.7.3 Reliability vs Validity

Reliability and validity are different but related. Reliability refers to the *consistency* of scores, whereas accuracy refers to the *accuracy* of scores. Validity depends on reliability. Reliability is necessary—but insufficient for—validity. That is, consistency is necessary—but insufficient for—accuracy. As depicted in Figure 8.14, a measure can be no more valid than it is reliable. A measure can be consistent but inaccurate; however, a measure cannot be accurate but inconsistent.

Figure 8.14 Reliability Versus Validity.

8.8 Conclusion

There are various types of research designs. Each type of research design differs in the extent to which it supports the ability to draw causal inferences (internal validity) versus the extent to which it supports the ability to identify processes that generalize to the real-world (external validity). In addition, it is important to understand the distinction between sample and population, and the distinction between mediation and moderation. It is also important to consider the levels of measurement used because they constrain the types of analyses that may be performed. In addition, it is important to consider the psychometrics of measurements, including multiple aspects of reliability (consistency) and validity (accuracy).

9

Basic Statistics

9.1 Getting Started

9.1.1 Load Packages

```
library("petersenlab")
library("DescTools")
library("pwr")
library("pwrss")
library("WebPower")
library("grid")
library("tidyverse")
```

9.2 Descriptive Statistics

Descriptive statistics are used to describe data. For instance, they may be used to describe the center, spread, or shape of the data. There are various indices of each.

9.2.1 Center

Indices to describe the *center* (central tendency) of a variable's data include:

- mean
- median
- Hodges-Lehmann statistic (aka pseudomedian)
- mode
- weighted mean

- weighted median

The mean of X (written as: $\bar{X}$) is calculated as in Equation 9.5:

$$\bar{X} = \frac{\sum X_i}{n} = \frac{X_1 + X_2 + ... + X_n}{n} \tag{9.1}$$

```
exampleValues <- c(0, 0, 10, 15, 20, 30, 1000)
exampleValues_mean <- apa(mean(exampleValues), 2)
```

That is, to compute the mean, sum all of the values and divide by the number of values (n). One issue with the mean is that it is sensitive to extreme (outlying) values. For instance, the mean of the values of 0, 0, 10, 15, 20, 30, and 1000 is 153.57.

```
exampleValues_median <- median(exampleValues)
```

The median is determined as the value at the 50th percentile (i.e., the value that is higher than 50% of the values and is lower than the other 50% of values). Compared to the mean, the median is less influenced by outliers. The median of the values of 0, 0, 10, 15, 20, 30, and 1000 is 15.

```
exampleValues_pseudomedian <- DescTools::HodgesLehmann(exampleValues)
```

The Hodges-Lehmann statistic (aka pseudomedian) is computed as the median of all pairwise means, and it is also robust to outliers. The pseudomedian of the values of 0, 0, 10, 15, 20, 30, and 1000 is 15.

```
exampleValues_mode <- petersenlab::Mode(exampleValues)
```

The mode is the most common/frequent value. The mode of the values of 0, 0, 10, 15, 20, 30, and 1000 is 0. The `petersenlab`[1] package (Petersen, 2024a) contains the `Mode()` function for computing the mode of a set of data.

If you want to give some values more weight to others, you can calculate a weighted mean and a weighted median (or other quantile), while assigning a weight to each value. The `petersenlab`[2] package (Petersen, 2024a) contains various functions for computing the weighted median (i.e., a weighted quantile at the 0.5 quantile, which is equivalent to the 50th percentile) based on Akinshin (2023). Because some projections are outliers, we use a trimmed version of the weighted Harrell-Davis quantile estimator for greater robustness.

Below is R code to estimate each:

[1] https://github.com/DevPsyLab/petersenlab
[2] https://github.com/DevPsyLab/petersenlab

```
#mean(data, na.rm = TRUE)
#median(data, na.rm = TRUE)
#DescTools::HodgesLehmann(exampleValues, na.rm = TRUE)
#petersenlab::Mode(exampleValues)
#weighted.mean(data, weights, na.rm = TRUE)
#petersenlab::wthdquantile(data, weights, probs = 0.5)
```

9.2.2 Spread

Indices to describe the *spread* (variability) of a variable's data include:

- standard deviation
- variance
- range
- minimum and maximum
- interquartile range (IQR)
- median absolute deviation

The (sample) variance of X (written as: s^2) is calculated as in Equation 9.2:

$$s^2 = \frac{\sum(X_i - \bar{X})^2}{n-1} \tag{9.2}$$

where X_i is each data point, $\bar{X}$ is the mean of X, and n is the number of data points.

The (sample) standard deviation of X (written as: s) is calculated as in Equation 9.3:

$$s = \sqrt{\frac{\sum(X_i - \bar{X})^2}{n-1}} \tag{9.3}$$

The range is calculated of X is calculated as in Equation 9.4:

$$\text{range} = \text{maximum} - \text{minimum} \tag{9.4}$$

The interquartile range (IQR) is calculated as in Equation 9.5:

$$\text{IQR} = Q_3 - Q_1 \tag{9.5}$$

where Q_3 is the score at the third quartile (i.e., 75th percentile), and Q_1 is the score at the first quartile (i.e., 25th percentile).

The median absolute deviation (MAD) is the median of all deviations from the median, and is calculated as in Equation 9.6:

$$\text{MAD} = \text{median}(|X_i - \tilde{X}|) \tag{9.6}$$

where $\tilde{X}$ is the median of `x`. Compared to the standard deviation, the median absolute deviation is more robust to outliers.

Below is R code to estimate each:

9.2.3 Shape

Indices to describe the *shape* of a variable's data include:

- skewness
- kurtosis

Below is R code to estimate each:

9.2.4 Combination

To estimate multiple indices of center, spread, and shape of the data, you can use the following code:

```
#psych::describe(mydata)

#mydata %>%
#  summarise(across(
#      everything(),
#      .fns = list(
#        n = ~ length(na.omit(.)),
#        missingness = ~ mean(is.na(.)) * 100,
#        M = ~ mean(., na.rm = TRUE),
#        SD = ~ sd(., na.rm = TRUE),
#        min = ~ min(., na.rm = TRUE),
#        max = ~ max(., na.rm = TRUE),
#        range = ~ max(., na.rm = TRUE) - min(., na.rm = TRUE),
#        IQR = ~ IQR(., na.rm = TRUE),
#        MAD = ~ mad(., na.rm = TRUE),
#        median = ~ median(., na.rm = TRUE),
#        pseudomedian = ~ DescTools::HodgesLehmann(., na.rm = TRUE),
#        mode = ~ petersenlab::Mode(., multipleModes = "mean"),
#        skewness = ~ psych::skew(., na.rm = TRUE),
```

```
#        kurtosis = ~ psych::kurtosi(., na.rm = TRUE)),
#      .names = "{.col}.{.fn}")) %>%
#    pivot_longer(
#      cols = everything(),
#      names_to = c("variable","index"),
#      names_sep = "\\.") %>%
#    pivot_wider(
#      names_from = index,
#      values_from = value)
```

9.3 Scores and Scales

There are many different types of scores and scales. This book focuses on raw scores and *z*-scores. For information on other scores and scales, including percentile ranks, *T*-scores, standard scores, scaled scores, and stanine scores, see here: https://isaactpetersen.github.io/Principles-Psychological-Assessment/scoresScales.html#scoreTransformation (Petersen, 2024c).

9.3.1 Raw Scores

Raw scores are the original data on the original metric. Thus, raw scores are considered *unstandardized*. For example, raw scores that represent the players' age may range from 20 to 40. Raw scores depend on the construct and unit; thus raw scores may not be comparable across variables.

9.3.2 *z* Scores

z scores have a mean of zero and a standard deviation of one. z scores are frequently used to render scores across variables more comparable. Thus, z scores are considered a form of a *standardized* score.

z scores are calculated using Equation 9.7:

$$z = \frac{X - \bar{X}}{\sigma} \tag{9.7}$$

where X is the observed score, $\bar{X}$ is the mean observed score, and σ is the standard deviation of the observed scores.

You can easily convert a variable to a z score using the `scale()` function:

```
scale(variable)
```

With a standard normal curve, 68% of scores fall within one standard deviation of the mean. 95% of scores fall within two standard deviations of the mean. 99.7% of scores fall within three standard deviations of the mean.

The area under a normal curve within one standard deviation of the mean is calculated below using the `pnorm()` function, which calculates the cumulative density function for a normal curve.

```
stdDeviations <- 1

pnorm(stdDeviations) - pnorm(stdDeviations * -1)
```

```
[1] 0.6826895
```

The area under a normal curve within one standard deviation of the mean is depicted in Figure 9.1.

```
x <- seq(-4, 4, length = 200)
y <- dnorm(x, mean = 0, sd = 1)
plot(x, y, type = "l",
     xlab = "z Score",
     ylab = "Normal Density")

x <- seq(stdDeviations * -1, stdDeviations, length = 100)
y <- dnorm(x, mean = 0, sd = 1)
polygon(c(stdDeviations * -1, x, stdDeviations),
        c(0, y, 0),
        col = "blue")
```

Figure 9.1 Density of Standard Normal Distribution. The blue region represents the area within one standard deviation of the mean.

The area under a normal curve within two standard deviations of the mean is calculated below:

```
stdDeviations <- 2

pnorm(stdDeviations) - pnorm(stdDeviations * -1)
```

```
[1] 0.9544997
```

The area under a normal curve within two standard deviations of the mean is depicted in Figure 9.2.

```
x <- seq(-4, 4, length = 200)
y <- dnorm(x, mean = 0, sd = 1)
plot(x, y, type = "l",
     xlab = "z Score",
     ylab = "Normal Density")

x <- seq(stdDeviations * -1, stdDeviations, length = 100)
y <- dnorm(x, mean = 0, sd = 1)
polygon(c(stdDeviations * -1, x, stdDeviations),
        c(0, y, 0),
        col = "blue")
```

Figure 9.2 Density of Standard Normal Distribution. The blue region represents the area within two standard deviations of the mean.

The area under a normal curve within three standard deviations of the mean is calculated below:

```
stdDeviations <- 3

pnorm(stdDeviations) - pnorm(stdDeviations * -1)
```

```
[1] 0.9973002
```

The area under a normal curve within three standard deviations of the mean is depicted in Figure 9.3.

```
x <- seq(-4, 4, length = 200)
y <- dnorm(x, mean = 0, sd = 1)
plot(x, y, type = "l",
     xlab = "z Score",
     ylab = "Normal Density")

x <- seq(stdDeviations * -1, stdDeviations, length = 100)
y <- dnorm(x, mean = 0, sd = 1)
polygon(c(stdDeviations * -1, x, stdDeviations),
        c(0, y, 0),
        col = "blue")
```

Figure 9.3 Density of Standard Normal Distribution. The blue region represents the area within three standard deviations of the mean.

If you want to determine the z score associated with a particular percentile in a normal distribution, you can use the `qnorm()` function. For instance, the z score associated with the 37th percentile is:

```
qnorm(.37)
```

```
[1] -0.3318533
```

9.4 Inferential Statistics

Inferential statistics are used to draw inferences regarding whether there is (a) a difference in level on variable across groups or (b) an association between variables. For instance, inferential statistics may be used to evaluate whether Quarterbacks tend to have longer careers compared to Running Backs. Or, they could be used to evaluate whether number of carries is associated with injury likelihood. To apply inferential statistics, we make use of the null hypothesis (H_0) and the alternative hypothesis (H_1).

9.4.1 Null Hypothesis Significance Testing

To draw statistical inferences, the frequentist statistics paradigm leverages null hypothesis significance testing. Frequentist statistics is the most widely used statistical paradigm. However, frequentist statistics is not the only statistical paradigm. Other statistical paradigms exist, including Bayesian statistics, which is based on Bayes' theorem. This chapter focuses on the frequentist approach to hypothesis testing, known as null hypothesis significance testing. We discuss Bayesian statistics in Chapter 16.

9.4.1.1 Null Hypothesis (H_0)

When testing whether there are differences in level across groups on a variable of interest, the null hypothesis (H_0) is that there is no difference in level across groups. For instance, when testing whether Quarterbacks tend to have longer careers compared to Running Backs, the null hypothesis (H_0) is that Quarterbacks do not systematically differ from Running Backs in the length of their career.

When testing whether there is an association between variables, the null hypothesis (H_0) is that there is no association between the variables. For instance, when testing whether number of carries is associated with injury likelihood, the null hypothesis (H_0) is that there is no association between number of carries and injury likelihood.

9.4.1.2 Alternative Hypothesis (H_1)

The alternative hypothesis (H_1) is the researcher's hypothesis that they want to evaluate. An alternative hypothesis (H_1) might be directional (i.e., one-sided) or non-directional (i.e., two-sided).

Directional hypotheses specify a particular direction, such as which group will have larger scores or which direction (positive or negative) two variables will be associated. Examples of directional hypotheses include:

- Quarterbacks have longer careers compared to Running Backs
- Number of carries is positively associated with injury likelihood

Non-directional hypotheses do not specify a particular direction. For instance, non-directional hypotheses may state that two groups differ but do not specify which group will have larger scores. Or, non-directional hypotheses may state that two variables are associated but do not state what the sign is of the association—i.e., positive or negative. Examples of non-directional hypotheses include:

- Quarterbacks differ in the length of their careers compared to Running Backs
- Number of carries is associated with injury likelihood

9.4.1.3 Statistical Significance

In science, statistical significance is evaluated with the p-value. The p-value does not represent the probability that you observed the result by chance. The p-value represents a conditional probability—it examines the probability of one event given another event. In particular, the p-value evaluates the likelihood that you would detect a result as at least as extreme as the one observed (in terms of the magnitude of the difference or of the association) given that the null hypothesis (H_0) is true.

This can be expressed in conditional probability notation, $P(A|B)$, which is the probability (likelihood) of event A occurring given that event B occurred (or given condition B).

The conditional probability notation for a left-tailed directional test (i.e., Quarterbacks have shorter careers than Running Backs; or number of carries is negatively associated with injury likelihood) is in Equation 9.8.

$$p\text{-value} = P(T \leq t|H_0) \tag{9.8}$$

where T is the test statistic of interest (e.g., the distribution of t-, $r-$, or F values, depending on the test) and t is the observed test statistic (e.g., t-, $r-$, or F-coefficient, depending on the test).

The conditional probability notation for a right-tailed directional test (i.e., Quarterbacks have longer careers than Running Backs; or number of carries is positively associated with injury likelihood) is in Equation 9.9.

$$p\text{-value} = P(T \geq t|H_0) \tag{9.9}$$

The conditional probability notation for a two-tailed non-directional test (i.e., Quarterbacks differ in the length of their careers compared to Running Backs; or number of carries is associated with injury likelihood) is in Equation 9.10.

$$p\text{-value} = 2 \times \min(P(T \leq t|H_0), P(T \geq t|H_0)) \tag{9.10}$$

where `min(a, b)` is the smaller number of `a` and `b`.

If the distribution of the test statistic is symmetric around zero, the p-value for the two-tailed non-directional test simplifies to Equation 9.11.

$$p\text{-value} = 2 \times P(T \geq |t||H_0) \tag{9.11}$$

Nevertheless, to be conservative (i.e., to avoid false positive/Type I errors), many researchers use two-tailed p-values regardless whether their hypothesis is one- or two-tailed.

For a test of group differences, the p-value evaluates the likelihood that you would observe a difference as large or larger than the one you observed between the groups if there were no systematic difference between the groups, as depicted in Figure 9.4. For instance, when evaluating whether Quarterbacks have longer careers than Running Backs, and you observed a mean difference of 0.03 years, the p-value evaluates the likelihood that you would observe a difference as larger or larger than 0.03 years between the groups if Quarterbacks do not differ from Running Backs in terms of the length of their career.

```
set.seed(52242)

nObserved <- 1000
nPopulation <- 1000000

observedGroups <- data.frame(
  score = c(rnorm(nObserved, mean = 47, sd = 3), rnorm(nObserved, mean = 52, sd = 3)),
  group = as.factor(c(rep("Group 1", nObserved), rep("Group 2", nObserved)))
)

populationGroups <- data.frame(
  score = c(rnorm(nPopulation, mean = 50, sd = 3.03), rnorm(nPopulation, mean = 50, sd = 3)),
  group = as.factor(c(rep("Group 1", nPopulation), rep("Group 2", nPopulation)))
)

ggplot2::ggplot(
  data = observedGroups,
  mapping = aes(
    x = score,
    fill = group,
    color = group
  )
) +
  geom_density(alpha = 0.5) +
  scale_color_manual(values = c("red", "blue")) +
  scale_fill_manual(values = c("red","blue")) +
  geom_vline(xintercept = mean(observedGroups$score[which(observedGroups$group == "Group 1")])) +
  geom_vline(xintercept = mean(observedGroups$score[which(observedGroups$group == "Group 2")])) +
  ggplot2::labs(
    x = "Score",
    y = "Frequency",
    title = "What is the probability my data would look like this..."
  ) +
```

```
  ggplot2::theme_classic(
    base_size = 16) +
  ggplot2::theme(
    legend.title = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank(),
    #plot.title.position = "plot"
    legend.position = "inside",
    legend.margin = margin(0, 0, 0, 0),
    legend.justification.top = "left",
    legend.justification.left = "top",
    legend.justification.bottom = "right",
    legend.justification.inside = c(1, 1),
    legend.location = "plot")

ggplot2::ggplot(
  data = populationGroups,
  mapping = aes(
    x = score,
    fill = group,
    color = group
  )
) +
  geom_density(alpha = 0.5) +
  scale_color_manual(values = c("red", "blue")) +
  scale_fill_manual(values = c("red","blue")) +
  geom_vline(xintercept = mean(populationGroups$score[which(populationGroups$group == "Group 1")])
  geom_vline(xintercept = mean(populationGroups$score[which(populationGroups$group == "Group 2")])
  ggplot2::labs(
    x = "Score",
    y = "Frequency",
    title = "...if in the population, the groups were really this:"
  ) +
  ggplot2::theme_classic(
    base_size = 16) +
  ggplot2::theme(
    legend.title = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank(),
    #plot.title.position = "plot",
```

```
    legend.position = "inside",
    legend.margin = margin(0, 0, 0, 0),
    legend.justification.top = "left",
    legend.justification.left = "top",
    legend.justification.bottom = "right",
    legend.justification.inside = c(1, 1),
    legend.location = "plot")
```

(a) What is the probability my data would look like this… **(b)** …if in the population, the groups were really this?

Figure 9.4 Interpretation of p-Values When Examining The Differences Between Groups. The vertical black lines reflect the group means.

For a test of whether two variables are associated, the p-value evaluates the likelihood that you would observe an association as strong or stronger than the one you observed between the groups if there were no association between the variables, as depicted in Figure 9.5. For instance, when evaluating whether number of carries is positively associated with injury likelihood, and you observed a correlation coefficient of $r = .25$ between number of carries and injury likelihood, the p-value evaluates the likelihood that you would observe a correlation as strong or stronger than $r = .25$ between the variables if number of carries is not associated with injury likelihood.

```
set.seed(52242)

observedCorrelation <- 0.9

correlations <- data.frame(criterion = rnorm(2000))
correlations$sample <- NA
correlations$sample[1:100] <- complement(correlations$criterion[1:100], observedCorrelation)
correlations$population <- complement(correlations$criterion, 0)

ggplot2::ggplot(
  data = correlations,
```

```
  mapping = aes(
    x = sample,
    y = criterion
  )
) +
  geom_point() +
  geom_smooth(method = "lm") +
  scale_x_continuous(
    limits = c(-3.5,3)
  ) +
  annotate(
    x = 0,
    y = 4,
    label = paste("italic(r) != ", 0, sep = ""),
    parse = TRUE,
    geom = "text",
    size = 7) +
  labs(
    x = "Predictor Variable",
    y = "Outcome Variable",
    title = "What is the probability my data would look like this..."
  ) +
  theme_classic(
    base_size = 16) +
  theme(
    legend.title = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank())

ggplot2::ggplot(
  data = correlations,
  mapping = aes(
    x = population,
    y = criterion
  )
) +
  geom_point() +
  geom_smooth(
    method = "lm",
    se = FALSE) +
  scale_x_continuous(
    limits = c(-2.5,2.5)
```

```
  ) +
  annotate(
    x = 0,
    y = 4,
    label = paste("italic(r) == '", "0.00", "'", sep = ""),
    parse = TRUE,
    geom = "text",
    size = 7) +
  labs(
    x = "Predictor Variable",
    y = "Outcome Variable",
    title = "...if in the population, the association was really this:"
  ) +
  theme_classic(
    base_size = 16) +
  theme(
    legend.title = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank())
```

(a) What is the probability my data would look like this…

(b) …if in the population, the association was really this?

Figure 9.5 Interpretation of p-Values When Examining The Association Between Variables.

Using what is called null-hypothesis significance testing (NHST), we consider an effect to be *statistically significant* if the p-value is less than some threshold, called the *alpha level*. In science, we typically want to be conservative because a false positive (i.e., Type I error) is considered more problematic than a false negative (i.e., Type II error). That is, we would rather say an effect does not exist when it really does than to say an effect does exist when it really does not. Thus, we typically set the alpha level to a low value, commonly .05.

Then, we would consider an effect to be *statistically significant* if the p-value is less than .05. That is, there is a small chance (5%; or 1 in 20 times) that we would observe an effect at least as extreme as the effect observed, if the null hypothesis were true. So, you might expect around 5% of tests where the null hypothesis is true to be statistically significant just by chance. We could lower the rate of Type II (i.e., false negative) errors—i.e., we could detect more effects—if we set the alpha level to a higher value (e.g., .10); however, raising the alpha level would raise the possibility of Type I (false positive) errors.

If the p-value is less than .05, we reject the null hypothesis (H_0) that there was no difference or association. Thus, we conclude that there was a statistically significant (non-zero) difference or association. If the p-value is greater than .05, we fail to reject the null hypothesis; the difference/association was not statistically significant. Thus, we do not have confidence that there was a difference or association. However, we do not accept the null hypothesis; it could be there we did not observe an effect because we did not have adequate power to detect the effect—e.g., if the effect size was small, the data were noisy, and the sample size was small and/or unrepresentative.

There are four general possibilities of decision making outcomes when performing null-hypothesis significance testing:

1. We (correctly) reject the null hypothesis when it is in fact false ($1-\beta$). This is a true positive. For instance, we may correctly determine that Quarterbacks have longer careers than Running Backs.
2. We (correctly) fail to reject the null hypothesis when it is in fact true ($1-\alpha$). This is a true negative. For instance, we may correctly determine that Quarterbacks do not have longer careers than Running Backs.
3. We (incorrectly) reject the null hypothesis when it is in fact true (α). This is a false positive. When performing null hypothesis testing, a false positive is known as a Type I error. For instance, we may incorrectly determine that Quarterbacks have longer careers than Running Backs when, in fact, Quarterbacks and Running Backs do not differ in their career length.
4. We (incorrectly) fail to reject the null hypothesis when it is in fact false (β). This is a false negative. When performing null hypothesis testing, a false negative is known as a Type II error. For instance, we may incorrectly determine that Quarterbacks and Running Backs do not differ in their career length when, in fact, Quarterbacks have longer careers than Running Backs.

A two-by-two confusion matrix for null-hypothesis significance testing is in Figure 9.6.

		Decision	
		Reject H_0	Fail to reject H_0
Truth	H_0 false	Correct True Positive 1 – β (“power”)	Type II error False Negative beta (β)
	H_0 true	Type I error False Positive alpha (α)	Correct True Negative 1 – α

Figure 9.6 A Two-by-Two Confusion Matrix for Null-Hypothesis Significance Testing.

In statistics, *power* is the probability of detecting an effect, if, in fact, the effect exists. Otherwise said, power is the probability of rejecting the null hypothesis, if, in fact, the null hypothesis is false. Power is influenced by several variables:

- the sample size (N): the larger the N, the greater the power
  - for group comparisons, the power depends on the sample size of each group
- the effect size: the larger the effect, the greater the power
  - for group comparisons, larger effect sizes reflect:
    - larger between-group variance, and
    - smaller within-group variance (i.e., strong measurement precision, i.e., reliability)
- the alpha level: the researcher specifies the alpha level (though it is typically set at .05); the higher the alpha level, the greater the power; however, the higher we set the alpha level, the higher the likelihood of Type I errors (false positives)
- one- versus two-tailed tests: one-tailed tests have higher power than two-tailed tests
- within-subject versus between-subject comparisons: within-subject designs tend to have greater power than between-subject designs

A plot of statistical power is in Figure 9.7.

```
m1 <- 0  # mu H0
sd1 <- 1.5 # sigma H0
m2 <- 3.5 # mu HA
```

```
sd2 <- 1.5 # sigma HA

z_crit <- qnorm(1-(0.05/2), m1, sd1)

# set length of tails
min1 <- m1-sd1*4
max1 <- m1+sd1*4
min2 <- m2-sd2*4
max2 <- m2+sd2*4
# create x sequence
x <- seq(min(min1,min2), max(max1, max2), .01)
# generate normal dist #1
y1 <- dnorm(x, m1, sd1)
# put in data frame
df1 <- data.frame("x" = x, "y" = y1)
# generate normal dist #2
y2 <- dnorm(x, m2, sd2)
# put in data frame
df2 <- data.frame("x" = x, "y" = y2)

# Alpha polygon
y.poly <- pmin(y1,y2)
poly1 <- data.frame(x=x, y=y.poly)
poly1 <- poly1[poly1$x >= z_crit, ]
poly1<-rbind(poly1, c(z_crit, 0))  # add lower-left corner

# Beta polygon
poly2 <- df2
poly2 <- poly2[poly2$x <= z_crit,]
poly2<-rbind(poly2, c(z_crit, 0))  # add lower-left corner

# power polygon; 1-beta
poly3 <- df2
poly3 <- poly3[poly3$x >= z_crit,]
poly3 <-rbind(poly3, c(z_crit, 0))  # add lower-left corner

# combine polygons.
poly1$id <- 3 # alpha, give it the highest number to make it the top layer
poly2$id <- 2 # beta
poly3$id <- 1 # power; 1 - beta
poly <- rbind(poly1, poly2, poly3)
poly$id <- factor(poly$id,  labels=c("power","beta","alpha"))

# plot with ggplot2
```

```
ggplot(poly, aes(x,y, fill=id, group=id)) +
  geom_polygon(show.legend=F, alpha=I(8/10)) +
  # add line for treatment group
  geom_line(data=df1, aes(x,y, color="H0", group=NULL, fill=NULL), linewidth=1.5, show_guide=F) +
  # add line for treatment group. These lines could be combined into one dataframe.
  geom_line(data=df2, aes(color="HA", group=NULL, fill=NULL),linewidth=1.5, show_guide=F) +
  # add vlines for z_crit
  geom_vline(xintercept = z_crit, linewidth=1, linetype="dashed") +
  # change colors
  scale_color_manual("Group",
                     values= c("HA" = "#981e0b","H0" = "black")) +
  scale_fill_manual("test", values= c("alpha" = "#0d6374","beta" = "#be805e","power"="#7cecee")) +
  # beta arrow
  annotate("segment", x=0.1, y=0.045, xend=1.3, yend=0.01, arrow = arrow(length = unit(0.3, "cm"))
  annotate("text", label="beta", x=0, y=0.05, parse=T, size=8) +
  # alpha arrow
  annotate("segment", x=4, y=0.043, xend=3.4, yend=0.01, arrow = arrow(length = unit(0.3, "cm")),
  annotate("text", label="frac(alpha,2)", x=4.2, y=0.05, parse=T, size=8) +
  # power arrow
  annotate("segment", x=6, y=0.2, xend=4.5, yend=0.15, arrow = arrow(length = unit(0.3, "cm")), li
  annotate("text", label=expression(paste(1-beta, "  (\"power\")")), x=6.1, y=0.21, parse=T, size=
  # H_0 title
  annotate("text", label="H[0]", x=m1, y=0.28, parse=T, size=8) +
  # H_a title
  annotate("text", label="H[1]", x=m2, y=0.28, parse=T, size=8) +
  ggtitle("Statistical Power") +
  # remove some elements
  theme(
    panel.grid.minor = element_blank(),
    panel.grid.major = element_blank(),
    panel.background = element_blank(),
    plot.background = element_rect(fill="white"),
    panel.border = element_blank(),
    axis.line = element_blank(),
    axis.text.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks = element_blank(),
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    plot.title = element_text(size=22))
```

Figure 9.7 Statistical Power (Adapted from Kristoffer Magnusson: https://rpsychologist.com/creating-a-typical-textbook-illustration-of-statistical-power-using-either-ggplot-or-base-graphics; archived at https://perma.cc/FG3J-85L6). The dashed line represents the critical value or threshold.

Interactive visualizations by Kristoffer Magnusson on p-values and null-hypothesis significance testing are below:

- https://rpsychologist.com/pvalue/ (archived at https://perma.cc/JP9F-9ZVY)
- https://rpsychologist.com/d3/pdist/ (archived at https://perma.cc/BE96-8LSJ)
- https://rpsychologist.com/d3/nhst/ (archived at https://perma.cc/ZU9A-37F3)

Twelve misconceptions about p-values (Goodman, 2008) are in Table 9.1.

Table 9.1 Twelve Misconceptions About p-Values from Goodman (2008). Goodman also provides a discussion about why each statement is false.

Number	Misconception
1	If $p = .05$, the null hypothesis has only a 5% chance of being true.
2	A nonsignificant difference (eg, $p > .05$) means there is no difference between groups.

Number	Misconception
3	A statistically significant finding is clinically important.
4	Studies with p-values on opposite sides of .05 are conflicting.
5	Studies with the same p-value provide the same evidence against the null hypothesis.
6	$p = .05$ means that we have observed data that would occur only 5% of the time under the null hypothesis.
7	$p = .05$ and $p < .05$ mean the same thing.
8	p-values are properly written as inequalities (e.g., "$p \leq .05$" when $p = .015$).
9	$p = .05$ means that if you reject the null hypothesis, the probability of a Type I error is only 5%.
10	With a $p = .05$ threshold for significance, the chance of a Type I error will be 5%.
11	You should use a one-sided p-value when you don't care about a result in one direction, or a difference in that direction is impossible.
12	A scientific conclusion or treatment policy should be based on whether or not the p-value is significant.

That is, the p-value is not:

- the probability that the effect was due to chance
- the probability that the null hypothesis is true
- the size of the effect
- the importance of the effect
- whether the effect is true, real, or causal

Statistical significance involves the *consistency* of an effect/association/difference; it suggests that the association/difference is reliably non-zero. However, just because something is statistically significant does not mean that it is important. For instance, consider that we discover that players who consume sports drink before a game tend to perform better than players who do not ($p < .05$). However, what if consumption of sports drinks is associated with an average improvement of 0.002 points per game. A small effect such as this might be detectable with a large sample size. This effect would be considered to be reliable/consistent because it is statistically significant. However, such an effect is so small that it results in differences that are not practically important. Thus, in addition to statistical significance, it is also important to consider practical significance.

9.4.2 Practical Significance

Practical significance deals with how large or important the effect/association/difference is. It is based on the magnitude of the effect, called the *effect size*. Effect size can be quantified in various ways including:

- Cohen's d
- Standardized regression coefficient (beta; β)
- Correlation coefficient (r)
- Cohen's ω (omega)
- Cohen's f
- Cohen's f^2
- Coefficient of determination (R^2)
- Eta squared (η^2)
- Partial eta squared (η_p^2)

9.4.2.1 Cohen's d

Cohen's d is calculated as in Equation 9.12:

$$\begin{aligned} d &= \frac{\text{mean difference}}{\text{pooled standard deviation}} \\ &= \frac{\bar{X}_1 - \bar{X}_2}{s} \end{aligned} \tag{9.12}$$

where:

$$s = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}} \tag{9.13}$$

where n_1 and n_2 is the sample size of group 1 and group 2, respectively, and s_1 and s_2 is the standard deviation of group 1 and group 2, respectively.

9.4.2.2 Standardized Regression Coefficient (Beta; β)

The standardized regression coefficient (beta; β) is used in multiple regression, and is calculated as in Equation 9.14:

$$\beta_x = B_x \times \frac{s_x}{s_y} \tag{9.14}$$

where B_x is the unstandardized regression coefficient of the predictor variable x in predicting the outcome variable y, s_x is the standard deviation of x, and s_y is the standard deviation of y.

9.4.2.3 Correlation Coefficient (r)

The formula for the correlation coefficient is in Chapter 10.

9.4.2.4 Cohen's ω

Cohen's ω is used in chi-square tests, and is calculated as in Equation 9.15:

$$\omega = \sqrt{\frac{\chi^2}{N} - \frac{df}{N}} \tag{9.15}$$

where χ^2 is the chi-square statistic from the test, N is the sample size, and df is the degrees of freedom.

9.4.2.5 Cohen's f

Cohen's f is commonly used in ANOVA, and is calculated as in Equation 9.16:

$$\begin{aligned} f &= \sqrt{\frac{R^2}{1-R^2}} \\ &= \sqrt{\frac{\eta^2}{1-\eta^2}} \end{aligned} \tag{9.16}$$

9.4.2.6 Cohen's f^2

Cohen's f^2 is commonly used in regression, and is calculated as in Equation 9.17:

$$\begin{aligned} f^2 &= \frac{R^2}{1-R^2} \\ &= \frac{\eta^2}{1-\eta^2} \end{aligned} \tag{9.17}$$

To calculate the effect size of a particular predictor, you can calculate Δf^2 as in Equation 9.18:

$$\begin{aligned} \Delta f^2 &= \frac{R^2_{\text{model}} - R^2_{\text{reduced}}}{1 - R^2_{\text{model}}} \\ &= \frac{\eta^2_{\text{model}} - \eta^2_{\text{reduced}}}{1 - \eta^2_{\text{model}}} \end{aligned} \tag{9.18}$$

where R^2_{model} is the R^2 of the model with the predictor variable of interest and R^2_{reduced} is the R^2 of the model without the predictor variable of interest.

9.4.2.7 Coefficient of Determination (R^2)

The coefficient of determination (R^2) reflects the proportion of variance in the outcome variable that is explained by the predictor variable(s). R^2 is commonly used in regression, and is calculated as in Equation 9.19:

$$\begin{aligned} R^2 &= 1 - \frac{\sum(Y_i - \hat{Y}_i)^2}{\sum(Y_i - \bar{Y})^2} \\ &= 1 - \frac{SS_{\text{residual}}}{SS_{\text{total}}} \\ &= 1 - \frac{\text{sum of squared residuals}}{\text{total sum of squares}} \\ &= \frac{f^2}{1+f^2} \\ &= \eta^2 \\ &= \frac{\text{variance explained in } Y}{\text{total variance in } Y} \end{aligned} \tag{9.19}$$

where Y_i is the observed value of the outcome variable for the ith observation, $\hat{Y}_i$ is the model predicted value for the ith observation, $\bar{Y}$ is the mean of the observed values of the outcome variable. The total sum of squares is an index of the total variation in the outcome variable.

9.4.2.8 Eta Squared (η^2) and Partial Eta Squared (η_p^2)

Like R^2, eta squared (η^2) reflects the proportion of variance in the dependent variable that is explained by the independent variable(s). η^2 is commonly used in ANOVA, and is calculated as in Equation 9.20:

$$\begin{aligned} \eta^2 &= \frac{SS_{\text{effect}}}{SS_{\text{total}}} \\ &= 1 - \frac{SS_{\text{residual}}}{SS_{\text{total}}} \\ &= 1 - \frac{\text{sum of squared residuals}}{\text{total sum of squares}} \\ &= \frac{f^2}{1+f^2} \\ &= R^2 \end{aligned} \tag{9.20}$$

where SS_{effect} is the sum of squares for the effect of interest and SS_{total} is the total sum of squares.

Partial eta squared (η_p^2) reflects the proportion of variance in the dependent variable that is explained by the independent variable while controlling for

the other independent variables. η_p^2 is commonly used in ANOVA, and is calculated as in Equation 9.21:

$$\eta_p^2 = \frac{SS_{\text{effect}}}{SS_{\text{effect}} + SS_{\text{error}}} \tag{9.21}$$

where SS_{effect} is the sum of squares for the effect of interest and SS_{error} is the sum of squares for the residual error term.

9.4.2.9 Effect Size Thresholds

Effect size thresholds (Cohen, 1988; McGrath & Meyer, 2006) for small, medium, and large effect sizes are in Table 9.2.

Table 9.2 Effect Size Thresholds for Small, Medium, and Large Effect Sizes.

Effect Size Index	Small	Medium	Large
Cohen's d	$\geq \lvert.20\rvert$	$\geq \lvert.50\rvert$	$\geq \lvert.80\rvert$
Standardized regression coefficient (beta; β)	$\geq \lvert.10\rvert$	$\geq \lvert.24\rvert$	$\geq \lvert.37\rvert$
Correlation coefficient (r)	$\geq \lvert.10\rvert$	$\geq \lvert.24\rvert$	$\geq \lvert.37\rvert$
Cohen's ω	$\geq .10$	$\geq .30$	$\geq .50$
Cohen's f	$\geq .10$	$\geq .25$	$\geq .40$
Cohen's f^2	$\geq .01$	$\geq .06$	$\geq .16$
Coefficient of determination (R^2)	$\geq .01$	$\geq .06$	$\geq .14$
Eta squared (η^2)	$\geq .01$	$\geq .06$	$\geq .14$
Partial eta squared (η_p^2)	$\geq .01$	$\geq .06$	$\geq .14$

9.5 Statistical Decision Tree

A statistical decision tree is a flowchart or decision tree that depicts which statistical test to use given the purpose of analysis, the type of data, etc. An example statistical decision tree is depicted in Figure 9.8.

This statistical decision tree can be generally summarized such that associations are examined with the correlation/regression family, and differences are examined with the t-test/ANOVA family, as depicted in Figure 9.9.

However, many statistical tests can be re-formulated in a regression framework, as in Figure 9.10.

Figure 9.8 A Statistical Decision Tree For Choosing an Appropriate Statistical Procedure. Adapted from: https://commons.wikimedia.org/wiki/File:InferentialStatisticalDecisionMakingTrees.pdf. The original source is: Corston, R. & Colman, A. M. (2000). *A crash course in SPSS for Windows*. Wiley-Blackwell. Changes were made to the original, including the addition of several statistical tests. *Note*: "Interval" as a level of measurement includes data with an "interval" or higher level of measurement; thus, it also includes data with a "ratio" level of measurement.

Both associations and differences can be examined with the regression family, which greatly simplifies our summary of the statistical decision tree, as depicted in Figure 9.11.

Thus, in most cases, the regression framework can be used to examine most questions regarding associations between variables or differences between groups.

For an online, interactive statistical decision tree to help you decide which statistical analysis to use, see here: https://www.statsflowchart.co.uk

Figure 9.9 Summary of A Statistical Decision Tree For Choosing an Appropriate Statistical Procedure.

9.6 Statistical Tests

9.6.1 *t*-Test

There are several *t*-tests:

- one-sample *t*-test
- two-samples *t*-test
  - independent samples *t*-test
  - paired samples *t*-test

A one-sample *t*-test is used to evaluate whether a sample mean differs systematically from a particular value. The null hypothesis is that the sample mean does not differ systematically from the pre-specified value. The alternative hypothesis is that the sample mean differs systematically from the pre-specified value. For instance, let's say you want to test out a new draft strategy. You could participate in a mock draft and draft players using the new strategy. Then, you could use a one-sample *t*-test to evaluate whether your new draft strategy yields players with more projected points than the average of players' projected points for other teams.

Two-samples *t*-tests are used to test for differences between scores of two groups. If the two groups are independent, the independent samples *t*-test is used. If the two groups involve paired samples, the paired samples *t*-test is

Figure 9.10 A Statistical Decision Tree For Choosing an Appropriate Statistical Procedure, Re-Formulated in a Regression Framework. Adapted from: https://commons.wikimedia.org/wiki/File:InferentialStatisticalDecisionMakingTrees.pdf. The original source is: Corston, R. & Colman, A. M. (2000). *A crash course in SPSS for Windows*. Wiley-Blackwell. Changes were made to the original, including re-formulating the tests in a regression framework.

used. The null hypothesis is that the mean of group 1 does not differ systematically from the mean of group 2. The alternative hypothesis is that the mean of group 1 differs systematically from the mean of group 2. For instance, you could use an independent-samples t-test if you want to examine whether Quarterbacks tend to have have longer careers than Running Backs. By contrast, you could use a paired samples t-test if you want to examine whether Quarterbacks tend to score more points in the second year of their contract compared to their rookie year, because the same subjects were assessed twice (i.e., a within-subject design).

Figure 9.11 Summary of A Statistical Decision Tree For Choosing an Appropriate Statistical Procedure.

9.6.2 Analysis of Variance

Analysis of variance (ANOVA) allows examining whether groups differ systematically as a function of one or more factors. There are multiple variants of ANOVA:

- one-way ANOVA
- factorial ANOVA
- repeated measures ANOVA (RM-ANOVA)
- multivariate ANOVA (MANOVA)

Like two-samples t-tests, ANOVA allows examining whether groups differ as a function of an independent variable. However, unlike a t-test, ANOVA allows examining multiple multiple independent variables and more than two groups. The null hypothesis is that the the groups' mean value does not differ systematically. The alternative hypothesis is that the groups' mean value differs systematically.

A one-way ANOVA examines whether two or more groups differ as a function of an independent variable. For instance, you could use a one-way ANOVA to evaluate if you want to evaluate whether multiple positions differ in their length of career. Factorial ANOVA examines whether two or more groups differ as a function of multiple independent variables. For instance, you could use factorial ANOVA to evaluate whether one's length of career depends on one's position and weight. Repeated measures ANOVA examines whether scores differ across repeated measures (e.g., across time) for the same participants.

For instance, you could use repeated-measures ANOVA to evaluate whether rookies score more points as the season progresses. Multivariate ANOVA examines whether multiple dependent variables differ as a function of one or more factor(s). For instance, you could use MANOVA to evaluate whether one's contract length and pay differ as a function of one's position.

9.6.3 Correlation

Correlation examines the association between a predictor and outcome variable. The null hypothesis is that the the two variables are not associated. The alternative hypothesis is that the two variables are associated.

The Pearson correlation coefficient (r) is calculated as in Equation 9.22:

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2 \sum(Y_i - \bar{Y})^2}} \tag{9.22}$$

where X is the predictor variable and Y is the outcome variable.

9.6.4 (Multiple) Regression

Regression, like correlation, examines the association between a predictor and outcome variable. However, unlike correlation, regression allows multiple predictor variables.

Regression with a single predictor takes the form in Equation 11.1. A regression line is depicted in Figure 11.4. Multiple regression (i.e., regression with multiple predictors) takes the form in Equation 11.2.

The null hypothesis is that the the predictor variable(s) are not associated with the outcome variable. The alternative hypothesis is that the predictor variable(s) are associated with the outcome variable.

9.6.5 Chi-Square Test

There are two primary types of chi-square tests:

- chi-square goodness-of-fit test
- chi-square test for association (aka test of independence)

The chi-square goodness-of-fit test evaluates whether a set of categorical data came from a specified distribution. The null hypothesis is that the data came

from the specified distribution. The alternative hypothesis is that the data did not come from the specified distribution.

The chi-square test for association evaluates whether two categorical variables are associated. The null hypothesis is that the two variables are not associated. The alternative hypothesis is that the two variables are associated.

9.6.6 Formulating Statistical Tests in Terms of Partitioned Variance

Many statistical tests can be formulated in terms of partitioned variance.

For instance, the t statistic from the independent-samples t-test and the F statistic from ANOVA can be thought of as the ratio of between-group variance to within-group variance, as in Equation 9.23:

$$t \text{ or } F = \frac{\text{between-group variance}}{\text{within-group variance}} \tag{9.23}$$

The correlation coefficient can be thought of as the ratio of shared variance (i.e., covariance) to total variance, as in Equation 9.24:

$$r = \frac{\text{shared variance}}{\text{total variance}} \tag{9.24}$$

The coefficient of determination (R^2) is the proportion of variance in the outcome variable that is explained by the predictor variables. η^2 is the proportion of variance in the dependent variable that is explained by the independent variables. The coefficient of determination and η^2 can be expressed as the ratio of variance explained in the outcome or dependent variable to the total variance in the outcome or dependent variable, as in Equation 9.25:

$$R^2 \text{ or } \eta^2 = \frac{\text{variance explained in the outcome variable}}{\text{total variance in the outcome variable}} \tag{9.25}$$

9.6.7 Critical Value

The critical value is the test value for a given test, above which the effect is considered to be statistically significant. The critical value for statistical significance for each test can be determined based on the degrees of freedom and alpha level. The degrees of freedom (df) refer to the number of values in the calculation of a test statistic that are free to vary.

```
alpha <- .05
N <- 200
nGroup1 <- 150
nGroup2 <- 150
numGroups <- 4
numLevelsFactorA <- 3
numLevelsFactorB <- 4
numMeasurements <- 4
numPredictors <- 5
numCategories <- 6
numRows <- 5
numColumns <- 2
```

9.6.7.1 One-Sample *t*-Test

For a one-sample t-test, the degrees of freedom is in Equation 9.26:

$$df = N - 1 \tag{9.26}$$

where N is sample size.

```
df_oneSampleTtest <- N - 1
```

One-tailed test:

```
qt(1 - alpha, df_oneSampleTtest)
```

```
[1] 1.652547
```

Two-tailed test:

```
qt(1 - alpha/2, df_oneSampleTtest)
```

```
[1] 1.971957
```

9.6.7.2 Independent-Samples *t*-Test

For an independent-samples t-test, the degrees of freedom is in Equation 9.27:

$$df = n_1 + n_2 - 2 \tag{9.27}$$

where n_1 is the sample size of group 1 and n_2 is the sample size of group 2.

```
df_independentSamplesTtest <- nGroup1 + nGroup2 - 2
```

One-tailed test:

```
qt(1 - alpha, df_independentSamplesTtest)
```

```
[1] 1.649983
```

Two-tailed test:

```
qt(1 - alpha/2, df_independentSamplesTtest)
```

```
[1] 1.967957
```

9.6.7.3 Paired-Samples *t*-Test

For a paired-samples t-test, the degrees of freedom is in Equation 9.28:

$$df = N - 1 \tag{9.28}$$

where N is sample size (i.e., the number of paired observations).

```
df_pairedSamplesTtest <- N - 1
```

One-tailed test:

```
qt(1 - alpha, df_pairedSamplesTtest)
```

```
[1] 1.652547
```

Two-tailed test:

```
qt(1 - alpha/2, df_pairedSamplesTtest)
```

```
[1] 1.971957
```

9.6.7.4 One-Way ANOVA

For a one-way ANOVA, the degrees of freedom is in Equation 9.29:

$$\begin{aligned} df_{\text{between}} &= g - 1 \\ df_{\text{within}} &= N - g \end{aligned} \tag{9.29}$$

where N is sample size and g is the number of groups.

```
df_betweenOneWayANOVA <- numGroups - 1
df_withinOneWayANOVA <- N - numGroups
```

One-tailed test:

```
qf(1 - alpha, df_betweenOneWayANOVA, df_withinOneWayANOVA)
```

```
[1] 2.650677
```

Two-tailed test:

```
qf(1 - alpha/2, df_betweenOneWayANOVA, df_withinOneWayANOVA)
```

```
[1] 3.183378
```

9.6.7.5 Factorial ANOVA

For a factorial two-way ANOVA, the degrees of freedom is in Equation 9.30:

$$\begin{aligned} df_{\text{Factor A}} &= a - 1 \\ df_{\text{Factor B}} &= b - 1 \\ df_{\text{Interaction}} &= (a-1)(b-1) \\ df_{\text{error}} &= ab(N-1) \end{aligned} \tag{9.30}$$

where N is sample size, a is the number of levels for factor A, and b is the number of levels for factor B.

```
df_factorA <- numLevelsFactorA - 1
df_factorB <- numLevelsFactorB - 1
df_interaction <- df_factorA * df_factorB
df_error <- numLevelsFactorA * numLevelsFactorB * (N - 1)
```

Factor A (one-tailed test):

```
qf(1 - alpha, df_factorA, df_error)
```

```
[1] 2.999494
```

Factor B (one-tailed test):

```
qf(1 - alpha, df_factorB, df_error)
```

```
[1] 2.608629
```

Interaction (one-tailed test):

```
qf(1 - alpha, df_interaction, df_error)
```

```
[1] 2.102376
```

Factor A (two-tailed test):

```
qf(1 - alpha/2, df_factorA, df_error)
```

```
[1] 3.694584
```

Factor B (two-tailed test):

```
qf(1 - alpha/2, df_factorB, df_error)
```

```
[1] 3.121587
```

Interaction (two-tailed test):

```
qf(1 - alpha/2, df_interaction, df_error)
```

```
[1] 2.413504
```

9.6.7.6 Repeated Measures ANOVA

For a repeated measures ANOVA, the degrees of freedom is in Equation 9.31:

$$\begin{aligned} df_1 &= T - 1 \\ df_2 &= (T-1)(N-1) \end{aligned} \tag{9.31}$$

where N is sample size and T is the number of measurements (i.e., the number of levels of the within-person factor: e.g., timepoints or conditions).

```
df1_RMANOVA <- numMeasurements - 1
df2_RMANOVA <- (numMeasurements - 1) * (N - 1)
```

One-tailed test:

```
qf(1 - alpha, df1_RMANOVA, df2_RMANOVA)
```

```
[1] 2.619828
```

Two-tailed test:

```
qf(1 - alpha/2, df1_RMANOVA, df2_RMANOVA)
```

```
[1] 3.138017
```

9.6.7.7 Correlation

For a correlation, the degrees of freedom is in Equation 9.32:

$$df = N - 2 \tag{9.32}$$

where N is sample size.

```
df_correlation <- N - 2
```

One-tailed test:

```
qt(1 - alpha, df_correlation)
```

```
[1] 1.652586
```

Two-tailed test:

```
qt(1 - alpha/2, df_correlation)
```

```
[1] 1.972017
```

9.6.7.8 Multiple Regression

For multiple regression, the degrees of freedom is in Equation 9.33:

$$\begin{aligned} df_1 &= p \\ df_2 &= N - p - 1 \end{aligned} \tag{9.33}$$

where N is sample size and p is the number of predictors.

```
df1_regression <- numPredictors
df2_regression <- N - numPredictors - 1
```

One-tailed test:

```
qf(1 - alpha, df1_regression, df2_regression)
```

```
[1] 2.260647
```

Two-tailed test:

```
qf(1 - alpha/2, df1_regression, df2_regression)
```

```
[1] 2.63243
```

9.6.7.9 Chi-Square Goodness-of-Fit Test

For the chi-square goodness-of-fit test, the degrees of freedom is in Equation 9.34:

$$df = c - 1 \tag{9.34}$$

where c is the number of categories.

```
df_chisquareGOF <- numCategories - 1
```

One-tailed test:

```
qchisq(1 - alpha, df_chisquareGOF)
```

```
[1] 11.0705
```

Two-tailed test:

```
qchisq(1 - alpha/2, df_chisquareGOF)
```

```
[1] 12.8325
```

9.6.7.10 Chi-Square Test for Association

For the chi-square test for association, the degrees of freedom is in Equation 9.35:

$$df = (r - 1) \times (c - 1) \tag{9.35}$$

where r is the number of rows in the contingency table and c is the number of columns in the contingency table.

```
df_chisquareAssociation <- (numRows - 1) * (numColumns - 1)
```

One-tailed test:

```
qchisq(1 - alpha, df_chisquareAssociation)
```

```
[1] 9.487729
```

Two-tailed test:

```
qchisq(1 - alpha/2, df_chisquareAssociation)
```

```
[1] 11.14329
```

9.6.8 Statistical Power

As described above, *statistical power* is the probability of detecting an effect, if, in fact, the effect exists. Statistical power for a given test can be calculated based on three factors:

- effect size
- sample size
- alpha level

Knowing any three of the following, you can calculate the fourth: statistical power, effect size, sample size, and alpha level. Below is `R` code for

calculating power for each of various statistical tests (i.e., a *power analysis*). For free point-and-click software for calculating statistical power, see G*Power: https://www.psychologie.hhu.de/arbeitsgruppen/allgemeine-psychologie-und-arbeitspsychologie/gpower.html

```
power <- .8
effectSize_d <- .5
effectSize_r <- .24
effectSize_beta <- .24
effectSize_f <- .25
effectSize_fSquared <- .06
effectSize_omega <- .3
```

When designing a study, it is important to consider power and the sample size needed to detect the hypothesized effect size. If your sample size is too small and you do not detect an effect (i.e., $p > .05$), you do not know whether your failure to detect the effect was because a) the effect does not exist, or b) the effect exists but you did not have enough power to detect it.

9.6.8.1 One-Sample *t*-Test

Solving for statistical power achieved (given effect size, sample size, and alpha level):

```
pwr::pwr.t.test(
  n = N,
  d = effectSize_d,
  sig.level = alpha,
  type = "one.sample",
  alternative = "two.sided")
```

```
     One-sample t test power calculation

              n = 200
              d = 0.5
      sig.level = 0.05
          power = 0.9999998
    alternative = two.sided
```

Solving for sample size needed (given effect size, power, and alpha level):

```r
pwr::pwr.t.test(
  power = power,
  d = effectSize_d,
  sig.level = alpha,
  type = "one.sample",
  alternative = "two.sided")
```

```
     One-sample t test power calculation

              n = 33.36713
              d = 0.5
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

Solving for the minimum detectable effect size (given sample size, power, and alpha level):

```r
pwr::pwr.t.test(
  power = power,
  n = N,
  sig.level = alpha,
  type = "one.sample",
  alternative = "two.sided")
```

```
     One-sample t test power calculation

              n = 200
              d = 0.1990655
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

9.6.8.2 Independent-Samples *t*-Test

9.6.8.2.1 Balanced Group Sizes

Solving for statistical power achieved (given effect size, sample size per group, and alpha level):

```r
pwr::pwr.t.test(
  n = N,
  d = effectSize_d,
  sig.level = alpha,
  type = "two.sample",
  alternative = "two.sided")
```

```
     Two-sample t test power calculation

              n = 200
              d = 0.5
      sig.level = 0.05
          power = 0.9987689
    alternative = two.sided

NOTE: n is number in *each* group
```

Solving for sample size per group needed (given effect size, power, and alpha level):

```r
pwr::pwr.t.test(
  power = power,
  d = effectSize_d,
  sig.level = alpha,
  type = "two.sample",
  alternative = "two.sided")
```

```
     Two-sample t test power calculation

              n = 63.76561
              d = 0.5
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

NOTE: n is number in *each* group
```

Solving for the minimum detectable effect size (given sample size per group, power, and alpha level):

```r
pwr::pwr.t.test(
  power = power,
  n = N,
  sig.level = alpha,
  type = "two.sample",
  alternative = "two.sided")
```

```
     Two-sample t test power calculation

              n = 200
              d = 0.2808267
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

NOTE: n is number in *each* group
```

9.6.8.2.2 Unbalanced Group Sizes

Solving for statistical power achieved (given effect size, sample size per group, and alpha level):

```r
pwr::pwr.t2n.test(
  n1 = nGroup1,
  n2 = nGroup2,
  d = effectSize_d,
  sig.level = alpha,
  alternative = "two.sided")
```

```
     t test power calculation

             n1 = 150
             n2 = 150
              d = 0.5
      sig.level = 0.05
          power = 0.9907677
    alternative = two.sided
```

Solving for sample size per group needed (given effect size, power, and alpha level):

```r
pwr::pwr.t2n.test(
  power = power,
  n1 = nGroup1,
  d = effectSize_d,
  sig.level = alpha,
  alternative = "two.sided")
```

```
     t test power calculation

             n1 = 150
             n2 = 40.22483
              d = 0.5
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

Solving for the minimum detectable effect size (given sample size per group, power, and alpha level):

```r
pwr::pwr.t2n.test(
  power = power,
  n1 = nGroup1,
  n2 = nGroup2,
  sig.level = alpha,
  alternative = "two.sided")
```

```
     t test power calculation

             n1 = 150
             n2 = 150
              d = 0.3245459
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

9.6.8.3 Paired-Samples *t*-Test

Solving for statistical power achieved (given effect size, sample size per group, and alpha level):

```r
pwr::pwr.t.test(
  n = N,
  d = effectSize_d,
  sig.level = alpha,
  type = "paired",
  alternative = "two.sided")
```

```
     Paired t test power calculation

              n = 200
              d = 0.5
      sig.level = 0.05
          power = 0.9999998
    alternative = two.sided

NOTE: n is number of *pairs*
```

Solving for sample size per group needed (given effect size, power, and alpha level):

```r
pwr::pwr.t.test(
  power = power,
  d = effectSize_d,
  sig.level = alpha,
  type = "paired",
  alternative = "two.sided")
```

```
     Paired t test power calculation

              n = 33.36713
              d = 0.5
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

NOTE: n is number of *pairs*
```

Solving for the minimum detectable effect size (given sample size per group, power, and alpha level):

```r
pwr::pwr.t.test(
  power = power,
  n = N,
  sig.level = alpha,
  type = "paired",
  alternative = "two.sided")
```

```
     Paired t test power calculation

              n = 200
              d = 0.1990655
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

NOTE: n is number of *pairs*
```

9.6.8.4 One-Way ANOVA

Solving for statistical power achieved (given effect size, sample size per group, and alpha level):

```r
pwr::pwr.anova.test(
  n = N,
  f = effectSize_f,
  sig.level = alpha,
  k = numGroups)
```

```
     Balanced one-way analysis of variance power calculation

              k = 4
              n = 200
              f = 0.25
      sig.level = 0.05
          power = 0.9999962

NOTE: n is number in each group
```

Solving for sample size per group needed (given effect size, power, and alpha level):

```r
pwr::pwr.anova.test(
  power = power,
  f = effectSize_f,
  sig.level = alpha,
  k = numGroups)
```

```
     Balanced one-way analysis of variance power calculation

              k = 4
              n = 44.59927
              f = 0.25
      sig.level = 0.05
          power = 0.8

NOTE: n is number in each group
```

Solving for the minimum detectable effect size (given sample size per group, power, and alpha level):

```r
pwr::pwr.anova.test(
  power = power,
  n = N,
  sig.level = alpha,
  k = numGroups)
```

```
     Balanced one-way analysis of variance power calculation

              k = 4
              n = 200
              f = 0.117038
      sig.level = 0.05
          power = 0.8

NOTE: n is number in each group
```

The power analysis code above assumes the groups are of equal size (i.e., a balanced design). If the design is unbalanced (i.e., there are different numbers of participants in each group), it may be necessary to conduct a power analysis via a simulation. Below is an example of evaluating the statistical power for detecting an effect unbalanced designs via simulation:

```r
nSim <- 1000 # number of simulations

# Function to generate data and perform ANOVA
simulate_anova <- function(nGroup1, nGroup2, f, alpha) {
  # Means for each group
  mean1 <- 0
  mean2 <- f * sqrt((nGroup1 + nGroup2) / 2)

  # Generate data
  group1 <- rnorm(nGroup1, mean = mean1, sd = 1)
  group2 <- rnorm(nGroup2, mean = mean2, sd = 1)

  # Combine data
  data <- data.frame(
    value = c(group1, group2),
    group = factor(rep(c("Group1", "Group2"), c(nGroup1, nGroup2)))
  )

  # Perform ANOVA
  aov_result <- aov(value ~ group, data = data)
  p_value <- summary(aov_result)[[1]][["Pr(>F)"]][1]

  # Check if p-value is less than alpha
  return(p_value < alpha)
}

# Run simulations
set.seed(52242) # for reproducibility
powerSimulationOneWayAnova <- replicate(
  nSim,
  simulate_anova(
    nGroup1 = 10,
    nGroup2 = 25,
    f = effectSize_f,
    alpha = alpha))

# Estimate power
mean(powerSimulationOneWayAnova)
```

```
[1] 0.774
```

9.6.8.5 Factorial ANOVA

The power analysis code below assumes the groups are of equal size (i.e., a balanced design). If the design is unbalanced (i.e., there are different numbers of participants in each group), it may be necessary to conduct a power analysis via a simulation. See Section 9.6.8.4 for an example power analysis simulation for one-way ANOVA.

Solving for statistical power achieved (given effect size, sample size per group, and alpha level):

```
pwr::pwr.anova.test(
  n = N,
  f = effectSize_f,
  sig.level = alpha,
  k = numLevelsFactorA)
```

```
     Balanced one-way analysis of variance power calculation

              k = 3
              n = 200
              f = 0.25
      sig.level = 0.05
          power = 0.9999238

NOTE: n is number in each group
```

```
pwr::pwr.anova.test(
  n = N,
  f = effectSize_f,
  sig.level = alpha,
  k = numLevelsFactorB)
```

```
     Balanced one-way analysis of variance power calculation

              k = 4
              n = 200
              f = 0.25
      sig.level = 0.05
          power = 0.9999962

NOTE: n is number in each group
```

```r
pwr::pwr.anova.test(
  n = N,
  f = effectSize_f,
  sig.level = alpha,
  k = numLevelsFactorA + numLevelsFactorB)
```

```
     Balanced one-way analysis of variance power calculation

              k = 7
              n = 200
              f = 0.25
      sig.level = 0.05
          power = 1

NOTE: n is number in each group
```

Solving for sample size per group needed (given effect size, power, and alpha level):

```r
pwr::pwr.anova.test(
  power = power,
  f = effectSize_f,
  sig.level = alpha,
  k = numLevelsFactorA)
```

```
     Balanced one-way analysis of variance power calculation

              k = 3
              n = 52.3966
              f = 0.25
      sig.level = 0.05
          power = 0.8

NOTE: n is number in each group
```

```r
pwr::pwr.anova.test(
  power = power,
  f = effectSize_f,
  sig.level = alpha,
  k = numLevelsFactorB)
```

```
     Balanced one-way analysis of variance power calculation

              k = 4
              n = 44.59927
              f = 0.25
      sig.level = 0.05
          power = 0.8

NOTE: n is number in each group
```

```
pwr::pwr.anova.test(
  power = power,
  f = effectSize_f,
  sig.level = alpha,
  k = numLevelsFactorA + numLevelsFactorB)
```

```
     Balanced one-way analysis of variance power calculation

              k = 7
              n = 32.05196
              f = 0.25
      sig.level = 0.05
          power = 0.8

NOTE: n is number in each group
```

Solving for the minimum detectable effect size (given sample size per group, power, and alpha level):

```
pwr::pwr.anova.test(
  power = power,
  n = N,
  sig.level = alpha,
  k = numLevelsFactorA)
```

```
     Balanced one-way analysis of variance power calculation

              k = 3
              n = 200
              f = 0.1270373
      sig.level = 0.05
```

```
          power = 0.8

NOTE: n is number in each group
```

```r
pwr::pwr.anova.test(
  power = power,
  n = N,
  sig.level = alpha,
  k = numLevelsFactorB)
```

```
     Balanced one-way analysis of variance power calculation

              k = 4
              n = 200
              f = 0.117038
      sig.level = 0.05
          power = 0.8

NOTE: n is number in each group
```

```r
pwr::pwr.anova.test(
  power = power,
  n = N,
  sig.level = alpha,
  k = numLevelsFactorA + numLevelsFactorB)
```

```
     Balanced one-way analysis of variance power calculation

              k = 7
              n = 200
              f = 0.09889082
      sig.level = 0.05
          power = 0.8

NOTE: n is number in each group
```

9.6.8.6 Repeated Measures ANOVA

Solving for statistical power achieved (given effect size, sample size per group, and alpha level):

```r
WebPower::wp.rmanova(
  n = N,
  ng = numGroups,
  nm = numMeasurements,
  f = effectSize_f,
  alpha = alpha,
  type = 0)
```

```
Repeated-measures ANOVA analysis

      n    f ng nm nscor alpha     power
    200 0.25  4  4     1  0.05 0.8484718

NOTE: Power analysis for between-effect test
URL: http://psychstat.org/rmanova
```

```r
WebPower::wp.rmanova(
  n = N,
  ng = numGroups,
  nm = numMeasurements,
  f = effectSize_f,
  alpha = alpha,
  type = 1)
```

```
Repeated-measures ANOVA analysis

      n    f ng nm nscor alpha     power
    200 0.25  4  4     1  0.05 0.8536292

NOTE: Power analysis for within-effect test
URL: http://psychstat.org/rmanova
```

```r
WebPower::wp.rmanova(
  n = N,
  ng = numGroups,
  nm = numMeasurements,
  f = effectSize_f,
  alpha = alpha,
  type = 2)
```

```
Repeated-measures ANOVA analysis

      n    f ng nm nscor alpha     power
```

```
    200 0.25  4  4     1  0.05 0.6756298

NOTE: Power analysis for interaction-effect test
URL: http://psychstat.org/rmanova
```

Solving for sample size per group needed (given effect size, power, and alpha level):

```r
WebPower::wp.rmanova(
  power = power,
  ng = numGroups,
  nm = numMeasurements,
  f = effectSize_f,
  alpha = alpha,
  type = 0)
```

```
Repeated-measures ANOVA analysis

           n    f ng nm nscor alpha power
    178.3971 0.25  4  4     1  0.05   0.8

NOTE: Power analysis for between-effect test
URL: http://psychstat.org/rmanova
```

```r
WebPower::wp.rmanova(
  power = power,
  ng = numGroups,
  nm = numMeasurements,
  f = effectSize_f,
  alpha = alpha,
  type = 1)
```

```
Repeated-measures ANOVA analysis

           n    f ng nm nscor alpha power
    175.7692 0.25  4  4     1  0.05   0.8

NOTE: Power analysis for within-effect test
URL: http://psychstat.org/rmanova
```

```r
WebPower::wp.rmanova(
  power = power,
  ng = numGroups,
```

```
  nm = numMeasurements,
  f = effectSize_f,
  alpha = alpha,
  type = 2)
```

```
Repeated-measures ANOVA analysis

           n    f ng nm nscor alpha power
    253.2369 0.25  4  4     1  0.05   0.8

NOTE: Power analysis for interaction-effect test
URL: http://psychstat.org/rmanova
```

Solving for the minimum detectable effect size (given sample size per group, power, and alpha level):

```
WebPower::wp.rmanova(
  power = power,
  n = N,
  ng = numGroups,
  nm = numMeasurements,
  alpha = alpha,
  type = 0)
```

```
Repeated-measures ANOVA analysis

      n         f ng nm nscor alpha power
    200 0.2358259  4  4     1  0.05   0.8

NOTE: Power analysis for between-effect test
URL: http://psychstat.org/rmanova
```

```
WebPower::wp.rmanova(
  power = power,
  n = N,
  ng = numGroups,
  nm = numMeasurements,
  alpha = alpha,
  type = 1)
```

```
Repeated-measures ANOVA analysis

      n         f ng nm nscor alpha power
```

```
    200 0.2342726  4  4     1  0.05   0.8

NOTE: Power analysis for within-effect test
URL: http://psychstat.org/rmanova
```

```
WebPower::wp.rmanova(
  power = power,
  n = N,
  ng = numGroups,
  nm = numMeasurements,
  alpha = alpha,
  type = 2)
```

```
Repeated-measures ANOVA analysis

      n         f ng nm nscor alpha power
    200 0.2817486  4  4     1  0.05   0.8

NOTE: Power analysis for interaction-effect test
URL: http://psychstat.org/rmanova
```

9.6.8.7 Correlation

Solving for statistical power achieved (given effect size, sample size per group, and alpha level):

```
pwr::pwr.r.test(
  n = N,
  r = effectSize_r,
  sig.level = alpha,
  alternative = "two.sided")
```

```
    approximate correlation power calculation (arctangh transformation)

              n = 200
              r = 0.24
      sig.level = 0.05
          power = 0.9310138
    alternative = two.sided
```

Solving for sample size per group needed (given effect size, power, and alpha level):

```r
pwr::pwr.r.test(
  power = power,
  r = effectSize_r,
  sig.level = alpha,
  alternative = "two.sided")
```

```
     approximate correlation power calculation (arctangh transformation)

              n = 133.1299
              r = 0.24
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

Solving for the minimum detectable effect size (given sample size per group, power, and alpha level):

```r
pwr::pwr.r.test(
  power = power,
  n = N,
  sig.level = alpha,
  alternative = "two.sided")
```

```
     approximate correlation power calculation (arctangh transformation)

              n = 200
              r = 0.1965767
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

9.6.8.8 Multiple Regression

Solving for statistical power achieved (given effect size, sample size, and alpha level):

```r
pwr::pwr.f2.test(
  f2 = effectSize_fSquared,
  sig.level = alpha,
  u = numPredictors,
  v = N - numPredictors - 1)
```

```
     Multiple regression power calculation

              u = 5
              v = 194
             f2 = 0.06
      sig.level = 0.05
          power = 0.7548031
```

```
pwrss::pwrss.t.reg(
  n = N,
  beta1 = effectSize_beta,
  k = numPredictors,
  alpha = alpha,
  alternative = "not equal")
```

```
 Linear Regression Coefficient (t Test)
 H0: beta1 = beta0
 HA: beta1 != beta0
 ------------------------------
  Statistical power = 0.936
  n = 200
 ------------------------------
 Alternative = "not equal"
 Degrees of freedom = 194
 Non-centrality parameter = 3.496
 Type I error rate = 0.05
 Type II error rate = 0.064
```

Solving for sample size needed (given effect size, power, and alpha level)—$v = N - \text{numberOfPredictors} - 1$; thus, $N = v + \text{numberOfPredictors} + 1$:

```
multipleRegressionSampleSizeModel <- pwr::pwr.f2.test(
  power = power,
  f2 = effectSize_fSquared,
  sig.level = alpha,
  u = numPredictors)

multipleRegressionSampleSizeModel
```

```
     Multiple regression power calculation

              u = 5
```

```
              v = 213.3947
             f2 = 0.06
      sig.level = 0.05
          power = 0.8
```

```
vNeeded <- multipleRegressionSampleSizeModel$v
sampleSizeNeeded <- vNeeded + numPredictors + 1
sampleSizeNeeded
```

```
[1] 219.3947
```

```
pwrss::pwrss.t.reg(
  power = power,
  beta1 = effectSize_beta,
  k = numPredictors,
  alpha = alpha,
  alternative = "not equal")
```

```
 Linear Regression Coefficient (t Test)
 H0: beta1 = beta0
 HA: beta1 != beta0
 ------------------------------
  Statistical power = 0.8
  n = 131
 ------------------------------
 Alternative = "not equal"
 Degrees of freedom = 124.427
 Non-centrality parameter = 2.823
 Type I error rate = 0.05
 Type II error rate = 0.2
```

Solving for the minimum detectable effect size (given sample size, power, and alpha level):

```
pwr::pwr.f2.test(
  power = power,
  sig.level = alpha,
  u = numPredictors,
  v = N - numPredictors - 1)
```

```
     Multiple regression power calculation
```

```
              u = 5
              v = 194
             f2 = 0.06597765
      sig.level = 0.05
          power = 0.8
```

9.6.8.9 Chi-Square Goodness-of-Fit Test

Solving for statistical power achieved (given effect size, sample size, and alpha level):

```r
pwr::pwr.chisq.test(
  N = N,
  w = effectSize_omega,
  df = numCategories - 1,
  sig.level = alpha)
```

```
     Chi squared power calculation

              w = 0.3
              N = 200
             df = 5
      sig.level = 0.05
          power = 0.9269225

NOTE: N is the number of observations
```

Solving for sample size needed (given effect size, power, and alpha level):

```r
pwr::pwr.chisq.test(
  power = power,
  w = effectSize_omega,
  df = numCategories - 1,
  sig.level = alpha)
```

```
     Chi squared power calculation

              w = 0.3
              N = 142.529
             df = 5
      sig.level = 0.05
```

```
          power = 0.8

NOTE: N is the number of observations
```

Solving for the minimum detectable effect size (given sample size, power, and alpha level):

```r
pwr::pwr.chisq.test(
  power = power,
  N = N,
  df = numCategories - 1,
  sig.level = alpha)
```

```
     Chi squared power calculation

              w = 0.2532543
              N = 200
             df = 5
      sig.level = 0.05
          power = 0.8

NOTE: N is the number of observations
```

9.6.8.10 Chi-Square Test for Association

Solving for statistical power achieved (given effect size, sample size, and alpha level):

```r
pwr::pwr.chisq.test(
  N = N,
  w = effectSize_omega,
  df = (numRows - 1)*(numColumns - 1),
  sig.level = alpha)
```

```
     Chi squared power calculation

              w = 0.3
              N = 200
             df = 4
      sig.level = 0.05
          power = 0.9431195

NOTE: N is the number of observations
```

Solving for sample size needed (given effect size, power, and alpha level):

```
pwr::pwr.chisq.test(
  power = power,
  w = effectSize_omega,
  df = (numRows - 1)*(numColumns - 1),
  sig.level = alpha)
```

```
     Chi squared power calculation

              w = 0.3
              N = 132.6143
             df = 4
      sig.level = 0.05
          power = 0.8

NOTE: N is the number of observations
```

Solving for the minimum detectable effect size (given sample size, power, and alpha level):

```
pwr::pwr.chisq.test(
  power = power,
  N = N,
  df = (numRows - 1)*(numColumns - 1),
  sig.level = alpha)
```

```
     Chi squared power calculation

              w = 0.2442875
              N = 200
             df = 4
      sig.level = 0.05
          power = 0.8

NOTE: N is the number of observations
```

9.6.8.11 Multilevel Modeling

Power analysis for multilevel modeling approaches is more complicated than it is for other statistical analyses, such as correlation, multiple regression, *t*-tests,

ANOVA[3], etc.

There are free web applications for calculating power in multilevel modeling:

- https://aguinis.shinyapps.io/ml_power/
- https://koumurayama.shinyapps.io/tmethod_mlm/
- https://webpower.psychstat.org/wiki/models/index

9.6.8.12 Path Analysis, Factor Analysis, and Structural Equation Modeling

Power analysis for latent variable modeling approaches like structural equation modeling (SEM) is more complicated than it is for other statistical analyses, such as correlation, multiple regression, *t*-tests, ANOVA[4], etc.

I provide an example of power analysis in SEM using Monte Carlo simulation in R here: https://isaactpetersen.github.io/Principles-Psychological-Assessment/sem.html#monteCarloPowerAnalysis (Petersen, 2024c).

There are also free web applications for calculating power in SEM:

- https://sjak.shinyapps.io/power4SEM/
- https://sempower.shinyapps.io/sempower/
- https://yilinandrewang.shinyapps.io/pwrSEM/
- https://webpower.psychstat.org/wiki/models/index

9.6.8.13 Mediation and Moderation

There are free tools for calculating power for tests of mediation and moderation:

- https://schoemanna.shinyapps.io/mc_power_med/
- https://www.causalevaluation.org/power-analysis.html (web application: https://powerupr.shinyapps.io/index/)
- https://webpower.psychstat.org/wiki/models/index

9.7 Conclusion

Descriptive statistics are used to describe the data, including the center, spread, or shape of data. Inferential statistics are used to draw inferences

[3] @sec-anova
[4] @sec-anova

regarding differences between groups or associations between variables. Null hypothesis signficance testing is a framework for inferential statistics, in which there is a null hypothesis and alternative hypothesis. The null hypothesis is that there is no difference between groups or that there is no association between variables. Statistical significance is evaluated with a p-value, which represents the probability of obtaining a result at least as extreme as the result observed if the null hypothesis is true. Effects with p-values less than .05 are considered statistically significant. However, it is also important to consider practical significance and effect sizes. When designing a study, it is important to consider statistical power and the sample size needed to detect the hypothesized effect size. You can determine a study's power based on the effect size, sample size, and alpha level.

10

Correlation Analysis

10.1 Getting Started

10.1.1 Load Packages

```
library("petersenlab")
library("XICOR")
library("tidyverse")
```

10.2 Overview of Correlation

Correlation is an index of the association between variables. Covariance is the association between variables and in an unstandardized metric that differs for variables with different scales. By contrast, correlation is in a standarized metric that does not differ for variables with different scales. When examining the association between variables that are interval or ratio levels of measurement, Pearson correlation is used. When examining the association between variables that are ordinal in level of measurement, Spearman correlation is used. Pearson correlation is an index of the *linear* association between variables. If a nonlinear association is present, other indices like xi [ξ; Chatterjee (2021)] and distance correlation coefficients are better suited to detect the association.

10.3 The Correlation Coefficient (r)

The formula for the correlation coefficient is in Equation 9.22.

The correlation coefficient ranges from −1.0 to +1.0. The correlation coefficient (r) tells you two things: (1) the direction (sign) of the association (positive or negative) and (2) the magnitude of the association. If the correlation coefficient is positive, the association is positive. If the correlation coefficient is negative, the association is negative. If the association is positive, as `X` increases, `Y` increases (or conversely, as `X` decreases, `Y` decreases). If the association is negative, as `X` increases, `Y` decreases (or conversely, as `X` decreases, `Y` increases). The smaller the absolute value of the correlation coefficient (i.e., the closer the r value is to zero), the weaker the association and the flatter the slope of the best-fit line in a scatterplot. The larger the absolute value of the correlation coefficient (i.e., the closer the absolute value of the r value is to one), the stronger the association and the steeper the slope of the best-fit line in a scatterplot. See Figure 10.1 for a range of different correlation coefficients and what some example data may look like for each direction and strength of association.

```
set.seed(52242)
correlations <- data.frame(criterion = rnorm(1000))

correlations$v1 <- complement(correlations$criterion, -1)
correlations$v2 <- complement(correlations$criterion, -.9)
correlations$v3 <- complement(correlations$criterion, -.8)
correlations$v4 <- complement(correlations$criterion, -.7)
correlations$v5 <- complement(correlations$criterion, -.6)
correlations$v6 <- complement(correlations$criterion, -.5)
correlations$v7 <- complement(correlations$criterion, -.4)
correlations$v8 <- complement(correlations$criterion, -.3)
correlations$v9 <- complement(correlations$criterion, -.2)
correlations$v10 <-complement(correlations$criterion, -.1)
correlations$v11 <-complement(correlations$criterion, 0)
correlations$v12 <-complement(correlations$criterion, .1)
correlations$v13 <-complement(correlations$criterion, .2)
correlations$v14 <-complement(correlations$criterion, .3)
correlations$v15 <-complement(correlations$criterion, .4)
correlations$v16 <-complement(correlations$criterion, .5)
correlations$v17 <-complement(correlations$criterion, .6)
correlations$v18 <-complement(correlations$criterion, .7)
correlations$v19 <-complement(correlations$criterion, .8)
correlations$v20 <-complement(correlations$criterion, .9)
correlations$v21 <-complement(correlations$criterion, 1)

par(mfrow = c(7,3), mar = c(1, 0, 1, 0))

# -1.0
plot(correlations$criterion, correlations$v1, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
```

```
     main = substitute(paste(italic(r), " = ", x, sep = ""), list(x = round(cor.test(x = correlati
abline(lm(v1 ~ criterion, data = correlations), col = "black")

# -.9
plot(correlations$criterion, correlations$v2, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = ""), list(x = round(cor.test(x = correlati
abline(lm(v2 ~ criterion, data = correlations), col = "black")

# -.8
plot(correlations$criterion, correlations$v3, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = ""), list(x = round(cor.test(x = correlati
abline(lm(v3 ~ criterion, data = correlations), col = "black")

# -.7
plot(correlations$criterion, correlations$v4, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = ""), list(x = round(cor.test(x = correlati
abline(lm(v4 ~ criterion, data = correlations), col = "black")

# -.6
plot(correlations$criterion, correlations$v5, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = ""), list(x = round(cor.test(x = correlati
abline(lm(v5 ~ criterion, data = correlations), col = "black")

# -.5
plot(correlations$criterion, correlations$v6, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = ""), list(x = round(cor.test(x = correlati
abline(lm(v6 ~ criterion, data = correlations), col = "black")

# -.4
plot(correlations$criterion, correlations$v7, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = ""), list(x = round(cor.test(x = correlati
abline(lm(v7 ~ criterion, data = correlations), col = "black")

# -.3
plot(correlations$criterion, correlations$v8, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = ""), list(x = round(cor.test(x = correlati
abline(lm(v8 ~ criterion, data = correlations), col = "black")

# -.2
plot(correlations$criterion, correlations$v9, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = ""), list(x = round(cor.test(x = correlati
abline(lm(v9 ~ criterion, data = correlations), col = "black")

# -.1
```

```
plot(correlations$criterion, correlations$v10, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = "")), list(x = round(cor.test(x = correlati
abline(lm(v10 ~ criterion, data = correlations), col = "black")

# 0.0
plot(correlations$criterion, correlations$v11, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = "")), list(x = round(cor.test(x = correlati
abline(lm(v11 ~ criterion, data = correlations), col = "black")

# 0.1
plot(correlations$criterion, correlations$v12, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = "")), list(x = round(cor.test(x = correlati
abline(lm(v12 ~ criterion, data = correlations), col = "black")

# 0.2
plot(correlations$criterion, correlations$v13, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = "")), list(x = round(cor.test(x = correlati
abline(lm(v13 ~ criterion, data = correlations), col = "black")

# 0.3
plot(correlations$criterion, correlations$v14, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = "")), list(x = round(cor.test(x = correlati
abline(lm(v14 ~ criterion, data = correlations), col = "black")

# 0.4
plot(correlations$criterion, correlations$v15, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = "")), list(x = round(cor.test(x = correlati
abline(lm(v15 ~ criterion, data = correlations), col = "black")

# 0.5
plot(correlations$criterion, correlations$v16, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = "")), list(x = round(cor.test(x = correlati
abline(lm(v16 ~ criterion, data = correlations), col = "black")

# 0.6
plot(correlations$criterion, correlations$v17, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = "")), list(x = round(cor.test(x = correlati
abline(lm(v17 ~ criterion, data = correlations), col = "black")

# 0.7
plot(correlations$criterion, correlations$v18, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = "")), list(x = round(cor.test(x = correlati
abline(lm(v18 ~ criterion, data = correlations), col = "black")
```

```
# 0.8
plot(correlations$criterion, correlations$v19, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = ""), list(x = round(cor.test(x = correlati
abline(lm(v19 ~ criterion, data = correlations), col = "black")

# 0.9
plot(correlations$criterion, correlations$v20, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = ""), list(x = round(cor.test(x = correlati
abline(lm(v20 ~ criterion, data = correlations), col = "black")

# 1.0
plot(correlations$criterion, correlations$v21, xaxt = "n", yaxt = "n", xlab = "" , ylab = "",
     main = substitute(paste(italic(r), " = ", x, sep = ""), list(x = round(cor.test(x = correlati
abline(lm(v21 ~ criterion, data = correlations), col = "black")

invisible(dev.off()) #par(mfrow = c(1,1))
```

Figure 10.1 Correlation Coefficients.

See Figure 10.2 for the interpretation of the magnitude and direction (sign) of various correlation coefficients.

```
library("patchwork")

set.seed(52242)
correlations2 <- data.frame(criterion = rnorm(15))

correlations2$v1 <- complement(correlations2$criterion, -1)
correlations2$v2 <- complement(correlations2$criterion, -.9)
correlations2$v3 <- complement(correlations2$criterion, -.8)
correlations2$v4 <- complement(correlations2$criterion, -.7)
correlations2$v5 <- complement(correlations2$criterion, -.6)
correlations2$v6 <- complement(correlations2$criterion, -.5)
correlations2$v7 <- complement(correlations2$criterion, -.4)
correlations2$v8 <- complement(correlations2$criterion, -.3)
correlations2$v9 <- complement(correlations2$criterion, -.2)
correlations2$v10 <-complement(correlations2$criterion, -.1)
correlations2$v11 <-complement(correlations2$criterion, 0)
correlations2$v12 <-complement(correlations2$criterion, .1)
correlations2$v13 <-complement(correlations2$criterion, .2)
correlations2$v14 <-complement(correlations2$criterion, .3)
correlations2$v15 <-complement(correlations2$criterion, .4)
correlations2$v16 <-complement(correlations2$criterion, .5)
correlations2$v17 <-complement(correlations2$criterion, .6)
correlations2$v18 <-complement(correlations2$criterion, .7)
correlations2$v19 <-complement(correlations2$criterion, .8)
correlations2$v20 <-complement(correlations2$criterion, .9)
correlations2$v21 <-complement(correlations2$criterion, 1)

# -1.0
p1 <- ggplot(
  data = correlations2,
  mapping = aes(
    x = criterion,
    y = v1
  )
) +
  geom_point() +
  geom_smooth(
    method = "lm",
    se = FALSE) +
  labs(
    title = "Perfect Negative Association",
    subtitle = expression(paste(italic("r"), " = ", "-1.0"))
```

```
  ) +
  theme_classic(
    base_size = 12) +
  theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.title.y = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank())

# -0.9
p2 <- ggplot(
  data = correlations2,
  mapping = aes(
    x = criterion,
    y = v2
  )
) +
  geom_point() +
  geom_smooth(
    method = "lm",
    se = FALSE) +
  labs(
    title = "Strong Negative Association",
    subtitle = expression(paste(italic("r"), " = ", "-.9"))
  ) +
  theme_classic(
    base_size = 12) +
  theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.title.y = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank())

# -0.5
p3 <- ggplot(
  data = correlations2,
  mapping = aes(
    x = criterion,
    y = v6
  )
```

```
) +
  geom_point() +
  geom_smooth(
    method = "lm",
    se = FALSE) +
  labs(
    title = "Moderate Negative Association",
    subtitle = expression(paste(italic("r"), " = ", "-.5"))
  ) +
  theme_classic(
    base_size = 12) +
  theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.title.y = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank())

# -0.2
p4 <- ggplot(
  data = correlations2,
  mapping = aes(
    x = criterion,
    y = v9
  )
) +
  geom_point() +
  geom_smooth(
    method = "lm",
    se = FALSE) +
  labs(
    title = "Weak Negative Association",
    subtitle = expression(paste(italic("r"), " = ", "-.2"))
  ) +
  theme_classic(
    base_size = 12) +
  theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.title.y = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank())
```

```
# 0.0
p5 <- ggplot(
  data = correlations2,
  mapping = aes(
    x = criterion,
    y = v11
  )
) +
  geom_point() +
  geom_smooth(
    method = "lm",
    se = FALSE) +
  labs(
    title = "No Association",
    subtitle = expression(paste(italic("r"), " = ", ".0"))
  ) +
  theme_classic(
    base_size = 12) +
  theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.title.y = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank())

# 0.2
p6 <- ggplot(
  data = correlations2,
  mapping = aes(
    x = criterion,
    y = v13
  )
) +
  geom_point() +
  geom_smooth(
    method = "lm",
    se = FALSE) +
  labs(
    title = "Weak Positive Association",
    subtitle = expression(paste(italic("r"), " = ", ".2"))
  ) +
  theme_classic(
    base_size = 12) +
```

```
  theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.title.y = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank())

# 0.5
p7 <- ggplot(
  data = correlations2,
  mapping = aes(
    x = criterion,
    y = v16
  )
) +
  geom_point() +
  geom_smooth(
    method = "lm",
    se = FALSE) +
  labs(
    title = "Moderate Positive Association",
    subtitle = expression(paste(italic("r"), " = ", ".5"))
  ) +
  theme_classic(
    base_size = 12) +
  theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.title.y = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank())

# 0.9
p8 <- ggplot(
  data = correlations2,
  mapping = aes(
    x = criterion,
    y = v20
  )
) +
  geom_point() +
  geom_smooth(
```

```
    method = "lm",
    se = FALSE) +
  labs(
    title = "Strong Positive Association",
    subtitle = expression(paste(italic("r"), " = ", ".9"))
  ) +
  theme_classic(
    base_size = 12) +
  theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.title.y = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank())

# 1.0
p9 <- ggplot(
  data = correlations2,
  mapping = aes(
    x = criterion,
    y = v21
  )
) +
  geom_point() +
  geom_smooth(
    method = "lm",
    se = FALSE) +
  labs(
    title = "Perfect Positive Association",
    subtitle = expression(paste(italic("r"), " = ", "1.0"))
  ) +
  theme_classic(
    base_size = 12) +
  theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.title.y = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank())

p1 + p2 + p3 + p4 + p5 + p6 + p7 + p8 + p9 +
  plot_layout(
```

```
    ncol = 3,
    heights = 1,
    widths = 1)
```

Figure 10.2 Interpretation of the Magnitude and Direction (Sign) of Correlation Coefficients.

Interactive visualizations by Kristoffer Magnusson on p-values and null-hypothesis significance testing are below:

- https://rpsychologist.com/correlation/ (archived at https://perma.cc/G8YR-VCM4)

10.4 Examples

10.4.1 Covariance

10.4.2 Pearson Correlation

10.4.3 Spearman Correlation

10.4.4 Nonlinear Correlation

10.4.5 Correlation Matrix

10.4.6 Correlogram

10.5 Impact of Outliers

10.6 Correlation Does Not Imply Causation

As described in Section 8.3.2.1, correlation does not imply causation. There are several reasons (described in Section 8.3.2.1) that, just because `X` is correlated with `Y` does not necessarily mean that `X` causes `Y`. However, correlation can still be useful. In order for two processes to be causally related, they must be associated. That is, association is necessary but insufficient for causality.

10.7 Conclusion

Correlation is an index of the association between variables. The correlation coefficient (r) ranges from -1 to $+1$, and indicates the sign and magnitude of the association. Although correlation does not imply causation, identifying associations between variables can still be useful because association is a necessary (but insufficient) condition for causality.

10.8 Session Info

11

Multiple Regression

11.1 Getting Started

11.1.1 Load Packages

```
library("petersenlab")
library("tidyverse")
library("knitr")
```

11.2 Overview of Multiple Regression

Multiple regression examines the association between multiple predictor variables and one outcome variable. It allows obtaining a more accurate estimate of the unique contribution of a given predictor variable, by controlling for other variables (covariates).

Regression with one predictor variable takes the form of Equation 11.1:

$$y = \beta_0 + \beta_1 x_1 + \epsilon \tag{11.1}$$

where y is the outcome variable, β_0 is the intercept, β_1 is the slope, x_1 is the predictor variable, and ϵ is the error term.

A regression line is depicted in Figure 11.4.

Regression with multiple predictors—i.e., multiple regression—takes the form of Equation 11.2:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \epsilon \tag{11.2}$$

where p is the number of predictor variables.

Figure 11.1 A Regression Best-Fit Line.

11.3 Components

- B = unstandardized coefficient: direction and magnitude of the estimate (original scale)
- β (beta) = standardized coefficient: direction and magnitude of the estimate (standard deviation scale)
- SE = standard error: uncertainty of unstandardized estimate

The unstandardized regression coefficient (B) is interpreted such that, for every unit change in the predictor variable, there is a ___ unit change in the outcome variable. For instance, when examining the association between age and fantasy points, if the unstandardized regression coefficient is 2.3, players score on average 2.3 more points for each additional year of age. (In reality, we might expect a nonlinear, inverted-U-shaped association between age and fantasy points such that players tend to reach their peak in the middle of their careers.) Unstandardized regression coefficients are tied to the metric of the raw data. Thus, a large unstandardized regression coefficient for two variables

may mean completely different things. Holding the strength of the association constant, you tend to see larger unstandardized regression coefficients for variables with smaller units and smaller unstandardized regression coefficients for variables with larger units.

Standardized regression coefficients can be obtained by standardizing the variables to z-scores so they all have a mean of zero and standard deviation of one. The standardized regression coefficient (β) is interpreted such that, for every standard deviation change in the predictor variable, there is a ___ standard deviation change in the outcome variable. For instance, when examining the association between age and fantasy points, if the standardized regression coefficient is 0.1, players score on average 0.1 standard deviation more points for each additional standard deviation of their year of age. Standardized regression coefficients—though not the case in all instances—tend to fall between [−1, 1]. Thus, standardized regression coefficients tend to be more comparable across variables and models compared to unstandardized regression coefficients. In this way, standardized regression coefficients provide a meaningful index of effect size.

11.4 Assumptions of Multiple Regression

Linear regression models make the following assumptions:

- there is a linear association between the predictor variables and the outcome variable
- there is homoscedasticity of the residuals; the residuals do not differ as a function of the predictor variables or as a function of the outcome variable
- the residuals are independent; they are uncorrelated with each other
- the residuals are normally distributed

11.5 Coefficient of Determination (R^2)

The coefficient of determination (R^2) reflects the proportion of variance in the outcome (dependent) variable that is explained by the model predictions: $R^2 = \frac{\text{variance explained in } Y}{\text{total variance in } Y}$. Various formulas for R^2 are in Equation 9.19. Larger R^2 values indicate greater accuracy. Multiple regression can be conceptualized with overlapping circles (similar to a venn diagram), where the

non-overlapping portions of the circles reflect nonshared variance and the overlapping portions of the circles reflect shared variance, as in Figure 11.4.

$$R^2 = \frac{\mathrm{D} + \mathrm{E} + \mathrm{G}}{\mathrm{A} + \mathrm{D} + \mathrm{E} + \mathrm{G}}$$

Figure 11.2 Conceptual Depiction of Proportion of Variance Explained (R^2) in an Outcome Variable (Y) by Multiple Predictors ($X1$ and $X2$) in Multiple Regression. The size of each circle represents the variable's variance. The proportion of variance in Y that is explained by the predictors is depicted by the areas in orange. The dark orange space (G) is where multiple predictors explain overlapping variance in the outcome. Overlapping variance that is explained in the outcome (G) will not be recovered in the regression coefficients when both predictors are included in the regression model. From Petersen (2024b) and Petersen (2024c).

One issue with R^2 is that it increases as the number of predictors increases, which can lead to overfitting if using R^2 as an index to compare models for purposes of selecting the "best-fitting" model. Consider the following example (adapted from Petersen (2024c)) in which you have one predictor variable and one outcome variable, as shown in Table 11.1.

Table 11.1 Example Data of Predictor (x1) and Outcome (y) Used for Regression Model.

y	x1
7	1
13	2
29	7
10	2

Using the data, the best fitting regression model is: $y = 3.98 + 3.59 \cdot x_1$. In this example, the R^2 is 0.98. The equation is not a perfect prediction, but with a single predictor variable, it captures the majority of the variance in the outcome.

Now consider the following example where you add a second predictor variable to the data above, as shown in Table 11.2.

Table 11.2 Example Data of Predictors (x1 and x2) and Outcome (y) Used for Regression Model.

y	x1	x2
7	1	3
13	2	5
29	7	1
10	2	2

With the second predictor variable, the best fitting regression model is: $y = 0.00 + 4.00 \cdot x_1 + 1.00 \cdot x_2$. In this example, the R^2 is 1.00. The equation with the second predictor variable provides a perfect prediction of the outcome.

Providing perfect prediction with the right set of predictor variables is the dream of multiple regression. So, using multiple regression, we often add predictor variables to incrementally improve prediction. Knowing how much variance would be accounted for by random chance follows Equation 11.3:

$$E(R^2) = \frac{K}{n-1} \tag{11.3}$$

where $E(R^2)$ is the expected value of R^2 (the proportion of variance explained), K is the number of predictor variables, and n is the sample size. The formula demonstrates that the more predictor variables in the regression model, the more variance will be accounted for by chance. With many predictor variables and a small sample, you can account for a large share of the variance merely by chance.

As an example, consider that we have 13 predictor variables to predict fantasy performance for 43 players. Assume that, with 13 predictor variables, we explain 38% of the variance ($R^2 = .38; r = .62$). We explained a lot of the variance in the outcome, but it is important to consider how much variance could have been explained by random chance: $E(R^2) = \frac{K}{n-1} = \frac{13}{43-1} = .31$. We expect to explain 31% of the variance, by chance, in the outcome. So, 82% of the variance explained was likely spurious (i.e., $\frac{.31}{.38} = .82$). As the sample size increases, the spuriousness decreases.

To account for the number of predictor variables in the model, we can use a modified version of R^2 called adjusted R^2 (R^2_{adj}). Adjusted R^2 (R^2_{adj}) accounts for the number of predictor variables in the model, based on how much would be expected to be accounted for by chance to penalize overfitting. Adjusted R^2 (R^2_{adj}) reflects the proportion of variance in the outcome (dependent) variable that is explained by the model predictions over and above what would be expected to be accounted for by chance, given the number of predictor variables in the model. The formula for adjusted R^2 (R^2_{adj}) is in Equation 11.4:

$$R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-p-1} \tag{11.4}$$

where p is the number of predictor variables in the model, and n is the sample size.

11.6 Overfitting

Statistical models applied to big data (e.g., data with many predictor variables) can *overfit* the data, which means that the statistical model accounts for error variance, which will not generalize to future samples. So, even though an overfitting statistical model appears to be accurate because it is accounting for more variance, it is not actually that accurate—it will predict new data less accurately than how accurately it accounts for the data with which the model was built. In the case of fantasy football analytics, this is especially relevant because there are hundreds if not thousands of variables we could consider for inclusion and many, many players when considering historical data.

Consider an example where you develop an algorithm to predict players' fantasy performance based on 2023 data using hundreds of predictor variables. To some extent, these predictor variables will likely account for true variance (i.e., signal) and error variance (i.e., noise). If we were to apply the same algorithm based on the 2023 prediction model to 2024 data, the prediction model would likely predict less accurately than with 2023 data. The regression coefficients in the

In Figure 11.3, the blue line represents the true distribution of the data, and the red line is an overfitting model:

```
set.seed(52242)

sampleSize <- 200
quadraticX <- rnorm(sampleSize)
quadraticY <- quadraticX ^ 2 + rnorm(sampleSize)
quadraticData <- cbind(quadraticX, quadraticY) %>%
  data.frame %>%
  arrange(quadraticX)

quadraticModel <- lm(
  quadraticY ~ quadraticX + I(quadraticX ^ 2),
  data = quadraticData)

quadraticNewData <- data.frame(
  quadraticX = seq(
    from = min(quadraticData$quadraticX),
    to = max(quadraticData$quadraticY),
    length.out = sampleSize))

quadraticNewData$quadraticY <- predict(
  quadraticModel,
  newdata = quadraticNewData)

loessFit <- loess(
  quadraticY ~ quadraticX,
  data = quadraticData,
  span = 0.01,
  degree = 1)

loessNewData <- data.frame(
  quadraticX = seq(
    from = min(quadraticData$quadraticX),
    to = max(quadraticData$quadraticY),
    length.out = sampleSize))

quadraticNewData$loessY <- predict(
  loessFit,
  newdata = quadraticNewData)

plot(
  x = quadraticData$quadraticX,
  y = quadraticData$quadraticY,
```

```
  xlab = "",
  ylab = "")

lines(
  quadraticNewData$quadraticY ~ quadraticNewData$quadraticX,
  lwd = 2,
  col = "blue")

lines(
  quadraticNewData$loessY ~ quadraticNewData$quadraticX,
  lwd = 2,
  col = "red")
```

Figure 11.3 Over-fitting Model in Red Relative to the True Distribution of the Data in Blue. From Petersen (2024b) and Petersen (2024c).

11.7 Covariates

Covariates are variables that you include in the statistical model to try to control for them so you can better isolate the unique contribution of the predictor variable(s) in relation to the outcome variable. Use of covariates examines the

association between the predictor variable and the outcome variable when holding people's level constant on the covariates. Inclusion of confounds as covariates allows potentially gaining a more accurate estimate of the causal effect of the predictor variable on the outcome variable. Ideally, you want to include any and all confounds as covariates. As described in Section 8.3.2.1, confounds are third variables that influence both the predictor variable and the outcome variable and explain their association. Covariates are potentially (but not necessarily) confounds. For instance, you might include the player's age as a covariate in a model that examines whether a player's 40-yard dash time at the NFL Combine predicts their fantasy points in their rookie year, but it may not be a confound.

11.8 Multicollinearity

Multicollinearity occurs when two or more predictor variables in a regression model are highly correlated. The problem of having multiple predictor variables that are highly correlated is that it makes it challenging to estimate the regression coefficients accurately.

Multicollinearity in multiple regression is depicted conceptually in Figure 11.4.

Consider the following example adapted from Petersen (2024c) where you have two predictor variables and one outcome variable, as shown in Table 11.3.

Table 11.3 Example Data of Predictors (x1 and x2) and Outcome (y) Used for Regression Model.

y	x1	x2
9	2.0	4
11	3.0	6
17	4.0	8
3	1.0	2
21	5.0	10
13	3.5	7

The second predictor variable is not very good—it is exactly twice the value of the first predictor variable; thus, the two predictor variables are perfectly correlated (i.e., $r = 1.0$). This means that there are different prediction equation possibilities that are equally good—see Equations in Equation 11.5:

Figure 11.4 Conceptual Depiction of Multicollinearity in Multiple Regression. From Petersen (2024b) and Petersen (2024c).

$$
\begin{aligned}
2x_2 &= y \\
0x_1 + 2x_2 &= y \\
4x_1 &= y \\
4x_1 + 0x_2 &= y \\
2x_1 + 1x_2 &= y \\
5x_1 - 0.5x_2 &= y \\
... &= y
\end{aligned}
\tag{11.5}
$$

Then, what are the regression coefficients? We do not know what are the correct regression coefficients because each of the possibilities fits the data equally well. Thus, when estimating the regression model, we could obtain arbitrary estimates of the regression coefficients with an enormous standard error around each estimate. In general, multicollinearity increases the uncertainty (i.e., standard errors and confidence intervals) around the parameter estimates. Any predictor variables that have a correlation above $\sim r = .30$ with each other could have an impact on the confidence interval of the regression coefficient. As the correlations among the predictor variables increase, the chance of getting an arbitrary answer increases, sometimes called "bouncing betas." So, it is important to examine a correlation matrix of the predictor variables before putting them in the same regression model. You can also

examine indices such as variance inflation factor (VIF).

To address multicollinearity, you can drop a redundant predictor or you can also use principal component analysis or factor analysis of the predictors to reduce the predictors down to a smaller number of meaningful predictors. For a meaningful answer in a regression framework that is precise and confident, you need a low level of intercorrelation among predictors, unless you have a very large sample size.

11.9 Impact of Oultiers

As with correlation, multiple regression can be strongly impacted by outliers.

11.10 Moderated Multiple Regression

When examining moderation in multiple regression, several steps are important:

- When computing the interaction term, first mean-center the predictor variables. Calculate the interaction term as the multiplication of the mean-centered predictor variables. Mean-centering the predictor variables when computing the interaction term is important for addressing issues regarding multicollinearity (Iacobucci et al., 2016).
- When including an interaction term in the model, make sure also to include the main effects.

11.11 Mediation

11.12 Bayesian Multiple Regression

11.13 Conclusion

Multiple regression allows examining the association between multiple predictor variables and one outcome variable. Inclusion of multiple predictors in the model allows for potentially greater predictive accuracy and identification of the extent to which each variable uniquely contributes to the outcome variable. As with correlation, an association does not imply causation. However, identifying associations is important because associations are a necessary (but insufficient) condition for causality. When developing a multiple regression model, it is important to pay attention for potential multicollinearity—it may become difficult to detect a given predictor variable as statistically significant due to the greater uncertainty around the parameter estimates.

12

Mixed Models

12.1 Getting Started

12.1.1 Load Packages

```
library("lme4")
library("lmerTest")
library("MuMIn")
library("emmeans")
library("sjstats")
library("mgcv")
library("AICcmodavg")
library("bbmle")
library("rstan")
library("brms")
library("cmdstanr") # todo: install.packages("cmdstanr", repos = c('https://stan-dev.r-universe.de
library("fitdistrplus")
library("parallel")
library("parallelly")
library("plotly")
library("viridis")
library("tidyverse")
```

12.1.2 Specify Package Options

```
emm_options(lmerTest.limit = 100000)
emm_options(pbkrtest.limit = 100000)
```

12.1.3 Load Data

```
load(file = "./data/nfl_depthCharts.RData")
load(file = "./data/player_stats_weekly.RData")
load(file = "./data/player_stats_seasonal.RData")
```

We created the `player_stats_weekly.RData` and `player_stats_seasonal.RData` objects in Section 4.4.3.

12.2 Overview of Mixed Models

We will discuss a modeling framework that goes by many terms, including mixed models, mixed-effects models, multilevel models, hierarchical linear models. They are sometimes called multilevel models and hierarchical linear models, whose name emphasizes the hierarchical structure of the data because the data are nonindependent. When observations (i.e., data points) are collected from multiple lower-level units (e.g., people) in an upper-level unit (e.g., married couple, family, classroom, school, neighborhood, team), the data from the lower-level units are considered "nested" within the upper-level unit. In this context, *nested data* refers to multiple observations from the same upper-level unit. For instance, longitudinal data are nested within the same participant. Students are nested within classrooms, which are nested within schools. Players are nested within teams.

When data are nested, the data from the lower-level unit are likely to be correlated, to some degree, because they come from the same upper-level unit. For example, multiple players may come from the same team, and the players' performance on that team is likely interrelated because they share common experiences and influence one another. Thus, data from multiple players on a given team are considered nested within that team. Longitudinal data can also be considered nested data, in which time points are nested within the person (i.e., the same player provides an observation across multiple time points). As we will discuss, it is important to account for levels of nesting when the observations are nonindependent.

These models are also sometimes called mixed models or mixed-effects models because the models can include a mix of fixed and random effects. Fixed effects are effects that are constant across individuals (i.e., upper-level units). Random effects are effects that vary across individuals (i.e., upper-level units). For instance, consider a longitudinal study of fantasy performance as a function of age. If we have longitudinal data for multiple players, the time points are

nested within players. Examining the association between age as a fixed effect in relation to fantasy performace would examine the assocication between a player's age and their fantasy performance while holding the association between age and fantasy performance constant across all players. That is, it would assume that all players show the same trajectory such as increase for 4 years then decrease. Examining the association between age as a random effect in relation to fantasy performace would examine the assocication between a player's age and their fantasy performance while allowing the association between age and fantasy performance to vary across players. That is, it would allow the possibility that some players improve with age, whereas other players decline in performance with age.

When including random effects of a variable (e.g., age) in a mixed model, it is also important to include fixed effects of that variable in the model. This is because random effects have a mean of zero. Fixed effects allow the mean to differ from zero. Thus, inclusion of random effects without the corresponding fixed effect can lead to bias in estimation of the association between the predictor variables and the outcome variable.

12.2.1 Ecological Fallacy

As described in Section 14.5.8, the ecological fallacy is the error of drawing inferences about an individual from group-level data. An type of ecological fallacy is Simpson's paradox.

12.2.2 Simpson's Paradox

Examples of Simpson's Paradox are depicted in Figures 13.5 and 12.1. In the example below, there is a positive association between the predictor variable and outcome variable for each group. However, when the groups are combined, there is a *negative* association between predictor variable and outcome variable. That is, the sign of an association can differ at different levels of analysis (e.g., group level versus person level).

Consider that we observe a between-person association between a predictor, for example, how much sports drink a player drinks and their performance, such that players who drink more sports drink before games tend to perform better during games. However, based on this association, if we draw the inference that sports drink consumption leads to better performance, this could be a faulty inference. It is possible that, at the *within*-person level, there is no association or even a negative association between sports drink consumption and performance. For helping to approximate causality, a much stronger test than relying on the between-person association would be to examine the association within the individual. That is, examining the association within

Figure 12.1 An Example of Simpson's Paradox. In this example, there is a positive association between x and y within each group (i.e., red or blue), but there is a negative association between x and y when the groups are combined. (Figure retrieved from https://en.wikipedia.org/wiki/File:Simpson%27s_paradox_continuous.svg)

the individual would examine: when a player consumes more sports drink, whether they perform better than when the same player consumes less sports drink. We describe within-person analyses to approximate causal inferences in Section 13.4.2.2.

In short, we often need to account for the groups or "levels" within which people are nested. When multiple observations are nested within the same upper-level unit, they are often correlated. That is, some variance is attributable to the lower-level unit (e.g., the player) and some variance is attributable to the upper-level unit (e.g., the team). It is important to evaluate how much variance is attributable to the upper-level unit. If substantial variance is attributable to the upper-level unit, it is important to account for the nested structure of the data. Mixed models are a useful approach to account for data with a nested structure so you can avoid committing the ecological fallacy.

12.3 Fantasy Points Per Season by Position, Age, and Experience

```
player_stats_seasonal_offense_subset <- player_stats_seasonal_offense %>%
  filter(position_group %in% c("QB","RB","WR","TE"))

player_stats_seasonal_offense_subset$position[which(player_stats_seasonal_offense_subset$position

player_stats_seasonal_kicking_subset <- player_stats_seasonal_kicking %>%
  filter(position == "K")

player_stats_seasonal_offense_subset <- bind_rows(
  player_stats_seasonal_offense_subset,
  player_stats_seasonal_kicking_subset
)

player_stats_seasonal_offense_subset$player_idFactor <- factor(player_stats_seasonal_offense_subse
player_stats_seasonal_offense_subset$positionFactor <- factor(player_stats_seasonal_offense_subset
```

```
seasons17week <- 2001:2020
seasons18week <- 2021:max(nfl_depthCharts$season, na.rm = TRUE)

endOfSeasonDepthCharts <- nfl_depthCharts %>%
  filter((season %in% seasons17week & week == 18) | (season %in% seasons18week & week == 19)) # ge

qb1s <- endOfSeasonDepthCharts %>%
  filter(position == "QB", depth_team == 1)

fb1s <- endOfSeasonDepthCharts %>%
  filter(position == "FB", depth_team == 1)

k1s <- endOfSeasonDepthCharts %>%
  filter(position == "K", depth_team == 1)

rb1s <- endOfSeasonDepthCharts %>%
  filter(position == "RB", depth_team == 1)

wr1s <- endOfSeasonDepthCharts %>%
  filter(position == "WR", depth_team == 1)

te1s <- endOfSeasonDepthCharts %>%
```

```
  filter(position == "TE", depth_team == 1)

player_stats_seasonal_offense_subsetDepth <- player_stats_seasonal_offense_subset %>%
  filter(player_id %in% c(
    qb1s$gsis_id,
    fb1s$gsis_id,
    k1s$gsis_id,
    rb1s$gsis_id,
    wr1s$gsis_id,
    te1s$gsis_id
    ))
```

Create a `newdata` object for generating the plots of model-implied fantasy points by age and position:

```
pointsPerSeason_positionAge_newData <- expand.grid(
  positionFactor = factor(c("FB","QB","RB","TE","WR")), #,"K"
  age = seq(from = 20, to = 40, length.out = 10000)
)

pointsPerSeason_positionAge_newData$ageCentered20 <- pointsPerSeason_positionAge_newData$age - 20
pointsPerSeason_positionAge_newData$ageCentered20Quadratic <- pointsPerSeason_positionAge_newData$
pointsPerSeason_positionAge_newData$years_of_experience <- floor(pointsPerSeason_positionAge_newDa
pointsPerSeason_positionAge_newData$years_of_experience[which(pointsPerSeason_positionAge_newData$
```

Create an object with complete cases for generating the plots of individuals' model-implied fantasy points by age and position:

```
player_stats_seasonal_offense_subsetCC <- player_stats_seasonal_offense_subset %>%
  filter(
    !is.na(player_idFactor),
    !is.na(fantasy_points),
    !is.na(positionFactor),
    !is.na(ageCentered20),
    !is.na(ageCentered20Quadratic),
    !is.na(years_of_experience))
```

12.3.1 Scatterplots of Fantasy Points by Age and Position

Scatterplots are a helpful tool for quickly examining the association between two variables. However, scatterplots—as well as correlation and multiple regression—can hide meaningful associations that differ across units of analysis.

12.3.1.1 Quarterbacks

A scatterplot of Quarterbacks' fantasy points by age is in Figure 12.2.

```
plot_scatterplotFantasyPointsByAgeQB <- ggplot(
  data = player_stats_seasonal_offense_subset %>%
    filter(position == "QB") %>%
    mutate(
      age = round(age, 2),
      fantasy_points = round(fantasy_points, 2)
    ),
  mapping = aes(
    x = age,
    y = fantasy_points,
    color = player_id)) +
  geom_point(
    aes(
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
  )) +
  geom_smooth(
    mapping = aes(
    x = age,
    y = fantasy_points),
    inherit.aes = FALSE
  ) +
  scale_color_viridis(discrete = TRUE) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age: Quarterbacks"
  ) +
  theme_classic() +
  theme(legend.position = "none")

ggplotly(
  plot_scatterplotFantasyPointsByAgeQB,
  tooltip = c("age","fantasy_points","text","label"))
```

Figure 12.2 Scatterplot of Fantasy Points by Age for Quarterbacks.

Based on the scatterplot (and the bivariate association below), Quarterbacks' fantasy points appear to increase with age.

```
cor.test(
  formula = ~ age + fantasy_points,
```

```
  data = player_stats_seasonal_offense_subset %>% filter(position == "QB")
)
```

```
    Pearson's product-moment correlation

data:  age and fantasy_points
t = 3.3424, df = 1096, p-value = 0.0008585
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.04153769 0.15866938
sample estimates:
      cor
0.1004516
```

12.3.1.2 Fullbacks

A scatterplot of Fullbacks' fantasy points by age is in Figure 12.3.

```
plot_scatterplotFantasyPointsByAgeFB <- ggplot(
  data = player_stats_seasonal_offense_subset %>%
    filter(position == "FB") %>%
    mutate(
      age = round(age, 2),
      fantasy_points = round(fantasy_points, 2)
    ),
  mapping = aes(
    x = age,
    y = fantasy_points,
    color = player_id)) +
  geom_point(
    aes(
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
  )) +
  geom_smooth(
    mapping = aes(
    x = age,
    y = fantasy_points),
    inherit.aes = FALSE
  ) +
  scale_color_viridis(discrete = TRUE) +
  labs(
    x = "Player Age (years)",
```

```
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age: Fullbacks"
  ) +
  theme_classic() +
  theme(legend.position = "none")

ggplotly(
  plot_scatterplotFantasyPointsByAgeFB,
  tooltip = c("age","fantasy_points","text","label"))
```

Figure 12.3 Scatterplot of Fantasy Points by Age for Fullbacks.

Based on the scatterplot, Fullbacks' fantasy points appear to be relatively stable across ages.

12.3.1.3 Running Backs

A scatterplot of Running Backs' fantasy points by age is in Figure 12.4.

```
plot_scatterplotFantasyPointsByAgeRB <- ggplot(
  data = player_stats_seasonal_offense_subset %>%
    filter(position == "RB") %>%
    mutate(
      age = round(age, 2),
      fantasy_points = round(fantasy_points, 2)
    ),
  mapping = aes(
    x = age,
    y = fantasy_points,
    color = player_id)) +
  geom_point(
    aes(
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
  )) +
  geom_smooth(
    mapping = aes(
    x = age,
    y = fantasy_points),
    inherit.aes = FALSE
  ) +
  scale_color_viridis(discrete = TRUE) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age: Running Backs"
  ) +
  theme_classic() +
  theme(legend.position = "none")

ggplotly(
  plot_scatterplotFantasyPointsByAgeRB,
  tooltip = c("age","fantasy_points","text","label"))
```

Figure 12.4 Scatterplot of Fantasy Points by Age for Running Backs.

Based on the scatterplot, Running Backs' fantasy points appear to be relatively stable across ages.

12.3.1.4 Wide Receivers

A scatterplot of Wide Receivers' fantasy points by age is in Figure 12.5.

```
plot_scatterplotFantasyPointsByAgeWR <- ggplot(
  data = player_stats_seasonal_offense_subset %>%
    filter(position == "WR") %>%
    mutate(
      age = round(age, 2),
      fantasy_points = round(fantasy_points, 2)
    ),
  mapping = aes(
    x = age,
    y = fantasy_points,
    color = player_id)) +
  geom_point(
    aes(
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
  )) +
  geom_smooth(
    mapping = aes(
    x = age,
    y = fantasy_points),
    inherit.aes = FALSE
  ) +
  scale_color_viridis(discrete = TRUE) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age: Wide Receivers"
  ) +
  theme_classic() +
  theme(legend.position = "none")

ggplotly(
  plot_scatterplotFantasyPointsByAgeWR,
  tooltip = c("age","fantasy_points","text","label"))
```

Figure 12.5 Scatterplot of Fantasy Points by Age for Wide Receivers.

Based on the scatterplot, Wide Receivers' fantasy points appear to be relatively stable across ages.

12.3.1.5 Tight Ends

A scatterplot of Tight Ends' fantasy points by age is in Figure 12.6.

```
plot_scatterplotFantasyPointsByAgeTE <- ggplot(
  data = player_stats_seasonal_offense_subset %>%
    filter(position == "TE") %>%
    mutate(
      age = round(age, 2),
      fantasy_points = round(fantasy_points, 2)
    ),
  mapping = aes(
    x = age,
    y = fantasy_points,
    color = player_id)) +
  geom_point(
    aes(
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
  )) +
  geom_smooth(
    mapping = aes(
    x = age,
    y = fantasy_points),
    inherit.aes = FALSE
  ) +
  scale_color_viridis(discrete = TRUE) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age: Tight Ends"
  ) +
  theme_classic() +
  theme(legend.position = "none")

ggplotly(
  plot_scatterplotFantasyPointsByAgeTE,
  tooltip = c("age","fantasy_points","text","label"))
```

Figure 12.6 Scatterplot of Fantasy Points by Age for Tight Ends.

Based on the scatterplot (and the bivariate association below), Tight Ends' fantasy points appear to increase with age.

```
cor.test(
  formula = ~ age + fantasy_points,
```

```
  data = player_stats_seasonal_offense_subset %>% filter(position == "TE")
)
```

```
    Pearson's product-moment correlation

data:  age and fantasy_points
t = 5.6618, df = 1435, p-value = 1.807e-08
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.09684768 0.19801648
sample estimates:
      cor
0.1478187
```

12.3.2 Plots of Raw Trajectories of Fantasy Points By Age and Player

Scatterplots can be helpful for quickly visualizing the association between two variables. However, as mentioned earlier, scatterplots can hide the association between variables at different units of analysis. For instance, consider that we are trying to predict how a player will perform based on their age. We are interested not only in what the association is between age and fantasy points between players (i.e., a between-person association). We are also interested in what the association is between age and fantasy points *within* a given player (and within each player; i.e., a within-individual association). Arguably, the within-individual association between age and fantasy points is more relevant to the prediction of performance than the association between age and fantasy points between players. Assuming that the between-player association between age and fantasy points is the same as the within-player association when it is not is an example of the ecological fallacy.

Below, we depict players' raw trajectories of fantasy points as a function of age. These are known as spaghetti plots. By examining the trajectory for each player, we can get a better understanding of hor performance changes (within an individual) as a function of age.

12.3.2.1 Quarterbacks

A plot of Quarterbacks' raw fantasy points data by age is in Figure 12.7.

```
plot_rawFantasyPointsByAgeQB <- ggplot(
  data = player_stats_seasonal_offense_subset %>%
```

```
    filter(position == "QB") %>%
    mutate(
      age = round(age, 2),
      fantasy_points = round(fantasy_points, 2)
    ),
  mapping = aes(
    x = age,
    y = fantasy_points,
    color = player_id)) +
  geom_line(
    aes(
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
  )) +
  scale_color_viridis(discrete = TRUE) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age: Quarterbacks"
  ) +
  theme_classic() +
  theme(legend.position = "none")

ggplotly(
  plot_rawFantasyPointsByAgeQB,
  tooltip = c("age","fantasy_points","text","label"))
```

Figure 12.7 Plot of Raw Trajectories of Fantasy Points by Age for Quarterbacks.

12.3.2.2 Fullbacks

A plot of Fullbacks' raw fantasy points data by age is in Figure 12.8.

```
plot_rawFantasyPointsByAgeFB <- ggplot(
  data = player_stats_seasonal_offense_subset %>%
    filter(position == "FB") %>%
    mutate(
      age = round(age, 2),
      fantasy_points = round(fantasy_points, 2)
    ),
  mapping = aes(
    x = age,
    y = fantasy_points,
    color = player_id)) +
  geom_line(
    aes(
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
  )) +
  scale_color_viridis(discrete = TRUE) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age: Fullbacks"
  ) +
  theme_classic() +
  theme(legend.position = "none")

ggplotly(
  plot_rawFantasyPointsByAgeFB,
  tooltip = c("age","fantasy_points","text","label"))
```

Figure 12.8 Plot of Raw Trajectories of Fantasy Points by Age for Fullbacks.

12.3.2.3 Running Backs

A plot of Running Backs' raw fantasy points data by age is in Figure 12.9.

```
plot_rawFantasyPointsByAgeRB <- ggplot(
  data = player_stats_seasonal_offense_subset %>%
    filter(position == "RB") %>%
    mutate(
      age = round(age, 2),
      fantasy_points = round(fantasy_points, 2)
    ),
  mapping = aes(
    x = age,
    y = fantasy_points,
    color = player_id)) +
  geom_line(
    aes(
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
  )) +
  scale_color_viridis(discrete = TRUE) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age: Running Backs"
  ) +
  theme_classic() +
  theme(legend.position = "none")

ggplotly(
  plot_rawFantasyPointsByAgeRB,
  tooltip = c("age","fantasy_points","text","label"))
```

Figure 12.9 Plot of Raw Trajectories of Fantasy Points by Age for Running Backs.

12.3.2.4 Wide Receivers

A plot of Wide Receivers' raw fantasy points data by age is in Figure 12.10.

```
plot_rawFantasyPointsByAgeWR <- ggplot(
  data = player_stats_seasonal_offense_subset %>%
    filter(position == "WR") %>%
    mutate(
      age = round(age, 2),
      fantasy_points = round(fantasy_points, 2)
    ),
  mapping = aes(
    x = age,
    y = fantasy_points,
    color = player_id)) +
  geom_line(
    aes(
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
  )) +
  scale_color_viridis(discrete = TRUE) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age: Wide Receivers"
  ) +
  theme_classic() +
  theme(legend.position = "none")

ggplotly(
  plot_rawFantasyPointsByAgeWR,
  tooltip = c("age","fantasy_points","text","label"))
```

Figure 12.10 Plot of Raw Trajectories of Fantasy Points by Age for Wide Receivers.

12.3.2.5 Tight Ends

A plot of Tight Ends' raw fantasy points data by age is in Figure 12.11.

```
plot_rawFantasyPointsByAgeTE <- ggplot(
  data = player_stats_seasonal_offense_subset %>%
    filter(position == "TE") %>%
    mutate(
      age = round(age, 2),
      fantasy_points = round(fantasy_points, 2)
    ),
  mapping = aes(
    x = age,
    y = fantasy_points,
    color = player_id)) +
  geom_line(
    aes(
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
  )) +
  scale_color_viridis(discrete = TRUE) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age: Tight Ends"
  ) +
  theme_classic() +
  theme(legend.position = "none")

ggplotly(
  plot_rawFantasyPointsByAgeTE,
  tooltip = c("age","fantasy_points","text","label"))
```

Figure 12.11 Plot of Raw Trajectories of Fantasy Points by Age for Tight Ends.

12.3.3 Linear Regression Models

12.3.3.1 Null Model

```
pointsPerSeason_nullModel <- lm(
  fantasy_points ~ 1,
  data = player_stats_seasonal_offense_subset,
  na.action = "na.exclude"
)

summary(pointsPerSeason_nullModel)
```

```
Call:
lm(formula = fantasy_points ~ 1, data = player_stats_seasonal_offense_subset,
    na.action = "na.exclude")

Residuals:
   Min     1Q Median     3Q    Max 
-68.13 -53.35 -29.35  28.75 418.61 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  60.8540     0.6321   96.27   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 73.53 on 13531 degrees of freedom
  (173 observations deleted due to missingness)
```

```
summary(pointsPerSeason_nullModel)$r.squared
```

```
[1] 0
```

```
AIC(pointsPerSeason_nullModel)
```

```
[1] 154719.7
```

```
MuMIn::AICc(pointsPerSeason_nullModel)
```

```
[1] 154719.7
```

A plot of the model-implied trajectories of fantasy points by age from the null model is in Figure 12.12.

```
pointsPerSeason_positionAge_newData$fantasyPoints_nullModel <- predict(
  object = pointsPerSeason_nullModel,
  newdata = pointsPerSeason_positionAge_newData
)

ggplot2::ggplot(
  data = pointsPerSeason_positionAge_newData,
  mapping = aes(
    x = age,
    y = fantasyPoints_nullModel
  )
) +
  geom_line(linewidth = 2) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position",
    subtitle = "Null Model"
  ) +
  theme_classic()
```

Figure 12.12 Plot of Model-Implied Trajectories of Fantasy Points by Age in Null Model.

12.3.3.2 Linear Model

```
pointsPerSeason_linearRegression <- lm(
  fantasy_points ~ positionFactor + ageCentered20 + positionFactor:ageCentered20,
  data = player_stats_seasonal_offense_subset,
  na.action = "na.exclude"
)

summary(pointsPerSeason_linearRegression)
```

```
Call:
lm(formula = fantasy_points ~ positionFactor + ageCentered20 +
   positionFactor:ageCentered20, data = player_stats_seasonal_offense_subset,
    na.action = "na.exclude")

Residuals:
    Min      1Q  Median      3Q     Max
-163.57  -51.04  -17.20   39.13  357.99

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                    10.8800    14.1652   0.768  0.44247
positionFactorQB               95.1482    14.9544   6.363 2.11e-10 ***
positionFactorRB               65.2732    14.7687   4.420 1.00e-05 ***
positionFactorTE               14.9087    14.8890   1.001  0.31671
positionFactorWR               42.0062    14.5667   2.884  0.00394 **
ageCentered20                   0.7586     1.9653   0.386  0.69952
positionFactorQB:ageCentered20  2.1526     2.0332   1.059  0.28976
positionFactorRB:ageCentered20 -0.7936     2.0836  -0.381  0.70331
positionFactorTE:ageCentered20  1.3635     2.0718   0.658  0.51047
positionFactorWR:ageCentered20  1.1458     2.0358   0.563  0.57358
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 75.01 on 6892 degrees of freedom
  (6803 observations deleted due to missingness)
Multiple R-squared:  0.1411,    Adjusted R-squared:   0.14
F-statistic: 125.8 on 9 and 6892 DF,  p-value: < 2.2e-16
```

```
summary(pointsPerSeason_linearRegression)$r.squared
```

```
[1] 0.1410917
```

```
AIC(pointsPerSeason_linearRegression)
```

```
[1] 79199.17
```

```
MuMIn::AICc(pointsPerSeason_linearRegression)
```

```
[1] 79199.21
```

A plot of the model-implied trajectories of fantasy points by age from the linear regression model is in Figure 12.13.

```
pointsPerSeason_positionAge_newData$fantasyPoints_linearRegression <- predict(
  object = pointsPerSeason_linearRegression,
  newdata = pointsPerSeason_positionAge_newData
)

ggplot2::ggplot(
  data = pointsPerSeason_positionAge_newData,
  mapping = aes(
    x = age,
    y = fantasyPoints_linearRegression,
    color = positionFactor
  )
) +
  geom_line(linewidth = 2) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position",
    subtitle = "Linear Regression Model",
    color = "Position"
  ) +
  theme_classic()
```

Figure 12.13 Plot of Model-Implied Trajectories of Fantasy Points by Age in Linear Regression Model.

12.3.3.3 Quadratic Model

```
pointsPerSeason_quadraticRegression <- lm(
  fantasy_points ~ positionFactor + ageCentered20 + ageCentered20Quadratic + positionFactor:ageCen
  data = player_stats_seasonal_offense_subset,
  na.action = "na.exclude"
)

summary(pointsPerSeason_quadraticRegression)
```

```
Call:
lm(formula = fantasy_points ~ positionFactor + ageCentered20 +
   ageCentered20Quadratic + positionFactor:ageCentered20 + positionFactor:ageCentered20Quadratic,
   data = player_stats_seasonal_offense_subset, na.action = "na.exclude")

Residuals:
    Min      1Q  Median      3Q     Max
-197.04  -50.94  -17.31   38.87  358.39

Coefficients:
```

```
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                         -3.5912   33.4748  -0.107   0.9146
positionFactorQB                   143.6064   34.6242   4.148  3.4e-05
positionFactorRB                    79.2210   34.3517   2.306   0.0211
positionFactorTE                    27.3585   34.5730   0.791   0.4288
positionFactorWR                    54.5244   34.0791   1.600   0.1097
ageCentered20                        5.2161    9.5488   0.546   0.5849
ageCentered20Quadratic              -0.2989    0.6267  -0.477   0.6334
positionFactorQB:ageCentered20     -11.1565    9.7573  -1.143   0.2529
positionFactorRB:ageCentered20      -5.0555    9.8760  -0.512   0.6087
positionFactorTE:ageCentered20      -2.4596    9.8439  -0.250   0.8027
positionFactorWR:ageCentered20      -2.6128    9.7578  -0.268   0.7889
positionFactorQB:ageCentered20Quadratic  0.7455   0.6343  1.175  0.2399
positionFactorRB:ageCentered20Quadratic  0.2843   0.6526  0.436  0.6631
positionFactorTE:ageCentered20Quadratic  0.2584   0.6437  0.401  0.6881
positionFactorWR:ageCentered20Quadratic  0.2486   0.6421  0.387  0.6986

(Intercept)
positionFactorQB                      ***
positionFactorRB                      *
positionFactorTE
positionFactorWR
ageCentered20
ageCentered20Quadratic
positionFactorQB:ageCentered20
positionFactorRB:ageCentered20
positionFactorTE:ageCentered20
positionFactorWR:ageCentered20
positionFactorQB:ageCentered20Quadratic
positionFactorRB:ageCentered20Quadratic
positionFactorTE:ageCentered20Quadratic
positionFactorWR:ageCentered20Quadratic
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 74.92 on 6887 degrees of freedom
  (6803 observations deleted due to missingness)
Multiple R-squared:  0.1437,    Adjusted R-squared:  0.142
F-statistic: 82.58 on 14 and 6887 DF,  p-value: < 2.2e-16
```

```r
summary(pointsPerSeason_quadraticRegression)$r.squared
```

```
[1] 0.1437412
```

```
AIC(pointsPerSeason_quadraticRegression)
```

```
[1] 79187.85
```

```
MuMIn::AICc(pointsPerSeason_quadraticRegression)
```

```
[1] 79187.93
```

A plot of the model-implied trajectories of fantasy points by age from the regression model with a quadratic term for age is in Figure 12.14.

```
pointsPerSeason_positionAge_newData$fantasyPoints_quadraticRegression <- predict(
  object = pointsPerSeason_quadraticRegression,
  newdata = pointsPerSeason_positionAge_newData
)

ggplot2::ggplot(
  data = pointsPerSeason_positionAge_newData,
  mapping = aes(
    x = age,
    y = fantasyPoints_quadraticRegression,
    color = positionFactor
  )
) +
  geom_line(linewidth = 2) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position",
    subtitle = "Quadratic Regression Model",
    color = "Position"
  ) +
  theme_classic()
```

Figure 12.14 Plot of Model-Implied Trajectories of Fantasy Points by Age in Quadratic Regression Model.

12.3.3.4 Compare Models

```
anova(
  pointsPerSeason_linearRegression,
  pointsPerSeason_quadraticRegression
)
```

```
Analysis of Variance Table

Model 1: fantasy_points ~ positionFactor + ageCentered20 + positionFactor:ageCentered20
Model 2: fantasy_points ~ positionFactor + ageCentered20 + ageCentered20Quadratic +
   positionFactor:ageCentered20 + positionFactor:ageCentered20Quadratic
  Res.Df      RSS Df Sum of Sq      F    Pr(>F)
1   6892 38776196
2   6887 38656580  5    119615 4.2621 0.0007164 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
AIC(
  pointsPerSeason_nullModel,
  pointsPerSeason_linearRegression,
```

```
  pointsPerSeason_quadraticRegression
  )
```

```
                                   df       AIC
pointsPerSeason_nullModel           2 154719.69
pointsPerSeason_linearRegression   11  79199.17
pointsPerSeason_quadraticRegression 16  79187.85
```

```
lmModels <- list(
  "nullModel" = pointsPerSeason_nullModel,
  "linearRegression" = pointsPerSeason_linearRegression,
  "quadraticRegression" = pointsPerSeason_quadraticRegression
)

bbmle::AICtab(lmModels)
```

```
                    dAIC    df
quadraticRegression     0.0 16
linearRegression       11.3 11
nullModel           75531.8 2
```

```
MuMIn::AICc(
  pointsPerSeason_nullModel,
  pointsPerSeason_linearRegression,
  pointsPerSeason_quadraticRegression
)
```

```
                                   df      AICc
pointsPerSeason_nullModel           2 154719.69
pointsPerSeason_linearRegression   11  79199.21
pointsPerSeason_quadraticRegression 16  79187.93
```

```
summary(pointsPerSeason_nullModel)$r.squared
```

```
[1] 0
```

```
summary(pointsPerSeason_linearRegression)$r.squared
```

```
[1] 0.1410917
```

```r
summary(pointsPerSeason_quadraticRegression)$r.squared
```

```
[1] 0.1437412
```

```r
deviance(pointsPerSeason_nullModel)
```

```
[1] 73167060
```

```r
deviance(pointsPerSeason_linearRegression)
```

```
[1] 38776196
```

```r
deviance(pointsPerSeason_quadraticRegression)
```

```
[1] 38656580
```

```r
logLik(pointsPerSeason_nullModel)
```

```
'log Lik.' -77357.85 (df=2)
```

```r
logLik(pointsPerSeason_linearRegression)
```

```
'log Lik.' -39588.59 (df=11)
```

```r
logLik(pointsPerSeason_quadraticRegression)
```

```
'log Lik.' -39577.93 (df=16)
```

12.3.4 Mixed Models

By accounting for which player each observation comes from using mixed models, we can examine the association between age and fantasy points in a more meaningful way, without violating the assumption in multiple regression that the observations are independent (i.e., that the residuals are uncorrelated).

12.3.4.1 Random Intercepts Model

```r
pointsPerSeason_randomIntercepts <- lmerTest::lmer(
  fantasy_points ~ 1 + (1 | player_idFactor),
  data = player_stats_seasonal_offense_subset,
  REML = FALSE,
  control = lmerControl(optimizer = "bobyqa")
)

summary(pointsPerSeason_randomIntercepts)
```

```
Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
  method [lmerModLmerTest]
Formula: fantasy_points ~ 1 + (1 | player_idFactor)
   Data: player_stats_seasonal_offense_subset
Control: lmerControl(optimizer = "bobyqa")

     AIC      BIC   logLik deviance df.resid
148044.0 148066.6 -74019.0 148038.0    13529

Scaled residuals:
    Min      1Q  Median      3Q     Max
-6.5321 -0.4328 -0.1682  0.3538  5.5810

Random effects:
 Groups          Name        Variance Std.Dev.
 player_idFactor (Intercept) 2197     46.87
 Residual                    2335     48.32
Number of obs: 13532, groups:  player_idFactor, 3358

Fixed effects:
             Estimate Std. Error        df t value Pr(>|t|)
(Intercept)   44.6942     0.9595 3834.9436   46.58   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
MuMIn::r.squaredGLMM(pointsPerSeason_randomIntercepts)
```

```
     R2m       R2c
[1,]   0 0.4847453
```

```r
performance::icc(pointsPerSeason_randomIntercepts)
```

```
# Intraclass Correlation Coefficient
```

```
    Adjusted ICC: 0.485
  Unadjusted ICC: 0.485
```

```
AIC(pointsPerSeason_randomIntercepts)
```

```
[1] 148044
```

```
AICc(pointsPerSeason_randomIntercepts)
```

```
numeric(0)
```

A plot of the model-implied trajectories of fantasy points by age from the mixed model with random intercepts is in Figure 12.15.

```
pointsPerSeason_positionAge_newData$fantasyPoints_randomIntercepts <- predict(
  object = pointsPerSeason_randomIntercepts,
  newdata = pointsPerSeason_positionAge_newData,
  re.form = NA
)

ggplot2::ggplot(
  data = pointsPerSeason_positionAge_newData,
  mapping = aes(
    x = age,
    y = fantasyPoints_randomIntercepts
  )
) +
  geom_line(linewidth = 2) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age",
    subtitle = "Random Intercepts Model"
  ) +
  theme_classic()
```

Figure 12.15 Plot of Model-Implied Trajectories of Fantasy Points by Age in Random Intercepts Mixed Model.

A plot of individuals' model-implied trajectories of fantasy points by age from the mixed model with random intercepts is in Figure 12.16.

```
player_stats_seasonal_offense_subsetCC$fantasyPoints_randomIntercepts <- predict(
  object = pointsPerSeason_randomIntercepts,
  newdata = player_stats_seasonal_offense_subsetCC
)

plot_individualFantasyPointsRandomIntercepts <- ggplot(
  data = player_stats_seasonal_offense_subsetCC %>%
    mutate(
      age = round(age, 2),
      fantasyPoints_randomIntercepts = round(fantasyPoints_randomIntercepts, 2)
    ),
  mapping = aes(
    x = age,
    y = fantasyPoints_randomIntercepts,
    group = player_id)) +
  geom_line(
    aes(
      x = age,
      y = fantasyPoints_randomIntercepts,
      text = player_display_name, # add player name for mouse over tooltip
```

```
      label = season # add season for mouse over tooltip
    ),
    linewidth = 0.5,
    color = "black") +
  geom_line(
    mapping = aes(
      x = age,
      y = fantasyPoints_randomIntercepts
    ),
    data = pointsPerSeason_positionAge_newData %>%
      mutate(
        age = round(age, 2),
        fantasy_points = round(fantasyPoints_randomIntercepts, 2)
        ),
    inherit.aes = FALSE,
    se = TRUE,
    color = "#3366FF",
    linewidth = 2
  ) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position: Random Intercepts Model",
    #color = "Position"
  ) +
  theme_classic()

ggplotly(
  plot_individualFantasyPointsRandomIntercepts,
  tooltip = c("age","fantasyPoints_randomIntercepts","text","label")
)
```

Figure 12.16 Plot of Individuals' Implied Trajectories of Fantasy Points by Age, from a Mixed Model with Random Intercepts. Overlaid with the Model-Implied Trajectory.

12.3.4.2 Random Intercepts Model with Position as Fixed-Effect Predictor

```r
pointsPerSeason_position <- lmerTest::lmer(
  fantasy_points ~ positionFactor + (1 | player_idFactor),
  data = player_stats_seasonal_offense_subset,
  REML = FALSE,
  control = lmerControl(optimizer = "bobyqa")
)

summary(pointsPerSeason_position)
```

```
Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
  method [lmerModLmerTest]
Formula: fantasy_points ~ positionFactor + (1 | player_idFactor)
   Data: player_stats_seasonal_offense_subset
Control: lmerControl(optimizer = "bobyqa")

     AIC      BIC   logLik deviance df.resid
147765.7 147818.2 -73875.8 147751.7    13525

Scaled residuals:
    Min      1Q  Median      3Q     Max
-6.5277 -0.4473 -0.1409  0.3592  5.5405

Random effects:
 Groups          Name        Variance Std.Dev.
 player_idFactor (Intercept) 1939     44.04
 Residual                    2336     48.33
Number of obs: 13532, groups:  player_idFactor, 3358

Fixed effects:
                 Estimate Std. Error       df t value Pr(>|t|)
(Intercept)        15.536      4.446 3449.346   3.494 0.000482 ***
positionFactorQB   61.412      5.179 3469.957  11.859  < 2e-16 ***
positionFactorRB   37.891      4.787 3499.420   7.916 3.26e-15 ***
positionFactorTE    9.898      4.903 3499.303   2.019 0.043592 *
positionFactorWR   27.227      4.692 3491.075   5.803 7.11e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) pstFQB pstFRB pstFTE
postnFctrQB -0.859
```

```
postnFctrRB -0.929  0.798
postnFctrTE -0.907  0.779  0.842
postnFctrWR -0.948  0.814  0.880  0.859
```

```r
MuMIn::r.squaredGLMM(pointsPerSeason_position)
```

```
            R2m       R2c
[1,] 0.06159295 0.4872378
```

```r
emmeans::emmeans(pointsPerSeason_position, "positionFactor")
```

```
 positionFactor emmean   SE   df lower.CL upper.CL
 FB               15.5 4.45 2895     6.81     24.3
 QB               76.9 2.66 2966    71.74     82.2
 RB               53.4 1.77 3241    49.95     56.9
 TE               25.4 2.07 3157    21.38     29.5
 WR               42.8 1.50 3287    39.82     45.7

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
performance::icc(pointsPerSeason_position)
```

```
# Intraclass Correlation Coefficient

    Adjusted ICC: 0.454
  Unadjusted ICC: 0.426
```

```r
AIC(pointsPerSeason_position)
```

```
[1] 147765.7
```

A plot of the model-implied trajectories of fantasy points by age and position from the mixed model with random intercepts and a fixed effect of position is in Figure 12.17.

```r
pointsPerSeason_positionAge_newData$fantasyPoints_position <- predict(
  object = pointsPerSeason_position,
  newdata = pointsPerSeason_positionAge_newData,
  re.form = NA
)
```

```
ggplot2::ggplot(
  data = pointsPerSeason_positionAge_newData,
  mapping = aes(
    x = age,
    y = fantasyPoints_position,
    color = positionFactor
  )
) +
  geom_line(linewidth = 2) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position",
    subtitle = "Random Intercepts Model With Position as Fixed-Effect Predictor",
    color = "Position"
  ) +
  theme_classic()
```

Figure 12.17 Plot of Model-Implied Trajectories of Fantasy Points by Age in Random Intercepts Mixed Model With Position as a Fixed-Effect Predictor.

A plot of individuals' model-implied trajectories of fantasy points by age and position from the mixed model with random intercepts and a fixed effect of position is in Figure 12.18.

```
player_stats_seasonal_offense_subsetCC$fantasyPoints_position <- predict(
  object = pointsPerSeason_position,
  newdata = player_stats_seasonal_offense_subsetCC
)

plot_individualFantasyPointsPosition <- ggplot(
  data = player_stats_seasonal_offense_subsetCC %>%
    mutate(
      age = round(age, 2),
      fantasyPoints_position = round(fantasyPoints_position, 2)
    ),
  mapping = aes(
    x = age,
    y = fantasyPoints_position,
    color = positionFactor,
    group = player_id)) +
  geom_line(
    aes(
      x = age,
      y = fantasyPoints_position,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    linewidth = 0.5) +
  geom_line(
    mapping = aes(
      x = age,
      y = fantasyPoints_position,
      color = positionFactor
    ),
    data = pointsPerSeason_positionAge_newData %>%
      mutate(
        age = round(age, 2),
        fantasyPoints_position = round(fantasyPoints_position, 2)
        ),
    inherit.aes = FALSE,
    se = TRUE,
    linewidth = 2
  ) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position:\nRandom Intercepts Model With Pos
    color = "Position"
```

```
  ) +
  theme_classic()

ggplotly(
  plot_individualFantasyPointsPosition,
  tooltip = c("age","fantasyPoints_position","text","label")
)
```

Figure 12.18 Plot of Individuals' Implied Trajectories of Fantasy Points by Age, from a Mixed Model With Random Intercepts and a Fixed-Effect of Position. Overlaid with the Model-Implied Trajectory by Position.

12.3.4.3 Identify the Best-Fitting Functional Form of Age

12.3.4.3.1 Linear Models

12.3.4.3.1.1 Random Intercepts, Fixed Linear Slopes

```r
pointsPerSeason_positionAgeFixedLinearSlopes <- lmerTest::lmer(
  fantasy_points ~ positionFactor + ageCentered20 + (1 | player_idFactor),
  data = player_stats_seasonal_offense_subset,
  REML = FALSE,
  control = lmerControl(optimizer = "bobyqa")
)

summary(pointsPerSeason_positionAgeFixedLinearSlopes)
```

```
Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
  method [lmerModLmerTest]
Formula:
fantasy_points ~ positionFactor + ageCentered20 + (1 | player_idFactor)
   Data: player_stats_seasonal_offense_subset
Control: lmerControl(optimizer = "bobyqa")

     AIC      BIC   logLik deviance df.resid
 76252.3  76307.0 -38118.1  76236.3     6894

Scaled residuals:
    Min      1Q  Median      3Q     Max
-6.3884 -0.4449 -0.1192  0.3879  4.5625

Random effects:
 Groups          Name        Variance Std.Dev.
 player_idFactor (Intercept) 2472     49.72
 Residual                    2633     51.31
Number of obs: 6902, groups:  player_idFactor, 1564

Fixed effects:
                  Estimate Std. Error        df t value Pr(>|t|)
(Intercept)        21.7839     8.6067 1650.2957   2.531 0.011465 *
positionFactorQB   89.3675     9.4485 1554.3915   9.458  < 2e-16 ***
positionFactorRB   46.7659     8.9701 1572.2982   5.214 2.10e-07 ***
```

```
positionFactorTE   16.4499     9.0641 1568.7893   1.815 0.069739 .
positionFactorWR   34.1349     8.8002 1569.5888   3.879 0.000109 ***
ageCentered20      -1.4746     0.2432 6431.9527  -6.062 1.42e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) pstFQB pstFRB pstFTE pstFWR
postnFctrQB -0.880
postnFctrRB -0.936  0.846
postnFctrTE -0.923  0.837  0.883
postnFctrWR -0.953  0.862  0.910  0.900
ageCentrd20 -0.179 -0.011  0.042  0.020  0.036
```

```r
MuMIn::r.squaredGLMM(pointsPerSeason_positionAgeFixedLinearSlopes)
```

```
            R2m       R2c
[1,] 0.09870768 0.5351623
```

```r
emmeans::emmeans(pointsPerSeason_positionAgeFixedLinearSlopes, "positionFactor")
```

```
 positionFactor emmean   SE   df lower.CL upper.CL
 FB               12.7 8.48 1392    -3.98     29.3
 QB              102.0 4.20 1395    93.79    110.3
 RB               59.4 2.96 1539    53.62     65.2
 TE               29.1 3.24 1496    22.76     35.5
 WR               46.8 2.39 1579    42.10     51.5

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
emmeans::emmeans(pointsPerSeason_positionAgeFixedLinearSlopes, "ageCentered20")
```

```
 ageCentered20 emmean   SE   df lower.CL upper.CL
          6.19     50 2.14 1418     45.8     54.2

Results are averaged over the levels of: positionFactor
Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
performance::icc(pointsPerSeason_positionAgeFixedLinearSlopes)
```

```
# Intraclass Correlation Coefficient

    Adjusted ICC: 0.484
  Unadjusted ICC: 0.436
```

```r
AIC(pointsPerSeason_positionAgeFixedLinearSlopes)
```

```
[1] 76252.29
```

A plot of the model-implied trajectories of fantasy points by age and position from the mixed model with random intercepts and fixed linear slopes is in Figure 12.19.

```r
pointsPerSeason_positionAge_newData$fantasyPoints_fixedLinearSlopes <- predict(
  object = pointsPerSeason_positionAgeFixedLinearSlopes,
  newdata = pointsPerSeason_positionAge_newData,
  re.form = NA
)

ggplot2::ggplot(
  data = pointsPerSeason_positionAge_newData,
  mapping = aes(
    x = age,
    y = fantasyPoints_fixedLinearSlopes,
    color = positionFactor
  )
) +
  geom_line(linewidth = 2) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position",
    subtitle = "Mixed Model with Random Intercepts and Fixed Linear Slopes",
    color = "Position"
  ) +
  theme_classic()
```

Figure 12.19 Plot of Model-Implied Trajectories of Fantasy Points by Age and Position in Mixed Model With Random Intercepts and Fixed Linear Slopes.

A plot of individuals model-implied trajectories of fantasy points by age and position from the mixed model with random intercepts and fixed linear slopes is in Figure 12.20.

```
player_stats_seasonal_offense_subsetCC$fantasyPoints_fixedLinearSlopes <- predict(
  object = pointsPerSeason_positionAgeFixedLinearSlopes,
  newdata = player_stats_seasonal_offense_subsetCC
)

plot_individualFantasyPointsFixedLinearSlopes <- ggplot(
  data = player_stats_seasonal_offense_subsetCC %>%
    mutate(
      age = round(age, 2),
      fantasyPoints_fixedLinearSlopes = round(fantasyPoints_fixedLinearSlopes, 2)
    ),
  mapping = aes(
    x = age,
    y = fantasyPoints_fixedLinearSlopes,
    color = positionFactor,
    group = player_id)) +
  geom_line(
    aes(
```

```
      x = age,
      y = fantasyPoints_fixedLinearSlopes,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    linewidth = 0.5) +
  geom_line(
    mapping = aes(
      x = age,
      y = fantasyPoints_fixedLinearSlopes,
      color = positionFactor
    ),
    data = pointsPerSeason_positionAge_newData,
    inherit.aes = FALSE,
    se = TRUE,
    linewidth = 2
  ) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position:\nModel With Random Intercepts and
    color = "Position"
  ) +
  theme_classic()

ggplotly(
  plot_individualFantasyPointsFixedLinearSlopes,
  tooltip = c("age","fantasyPoints_fixedLinearSlopes","text","label")
)
```

Figure 12.20 Plot of Individuals' Implied Trajectories of Fantasy Points by Age and Position, from a Mixed Model With Random Intercepts and Fixed Slopes. Overlaid with the Model-Implied Trajectory by Position.

12.3.4.3.1.2 Random Intercepts, Random Linear Slopes

```r
pointsPerSeason_positionAgeRandomLinearSlopes <- lmerTest::lmer(
  fantasy_points ~ positionFactor + ageCentered20 + (1 + ageCentered20 | player_idFactor),
  data = player_stats_seasonal_offense_subset,
  REML = FALSE,
  control = lmerControl(optimizer = "bobyqa")
)

summary(pointsPerSeason_positionAgeRandomLinearSlopes)
```

```
Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
  method [lmerModLmerTest]
Formula:
fantasy_points ~ positionFactor + ageCentered20 + (1 + ageCentered20 |
    player_idFactor)
   Data: player_stats_seasonal_offense_subset
Control: lmerControl(optimizer = "bobyqa")

     AIC      BIC   logLik deviance df.resid
 76022.6  76091.0 -38001.3  76002.6     6892

Scaled residuals:
    Min      1Q  Median      3Q     Max
-6.4634 -0.4333 -0.1198  0.3763  4.7521

Random effects:
 Groups          Name          Variance Std.Dev. Corr
 player_idFactor (Intercept)   3308.97  57.524
                 ageCentered20   27.62   5.256   -0.51
 Residual                      2390.84  48.896
Number of obs: 6902, groups:  player_idFactor, 1564

Fixed effects:
                  Estimate Std. Error        df t value Pr(>|t|)
(Intercept)        24.7285     8.6754 1648.4577   2.850 0.004420 **
positionFactorQB   88.4762     9.4228 1487.8408   9.390  < 2e-16 ***
positionFactorRB   46.4788     8.9397 1513.4085   5.199 2.28e-07 ***
positionFactorTE   15.9189     9.0297 1501.3250   1.763 0.078112 .
positionFactorWR   33.7757     8.7691 1501.8626   3.852 0.000122 ***
ageCentered20      -2.0306     0.3367  761.3446  -6.031 2.54e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) pstFQB pstFRB pstFTE pstFWR
```

```
postnFctrQB -0.873
postnFctrRB -0.929  0.845
postnFctrTE -0.915  0.837  0.883
postnFctrWR -0.946  0.862  0.910  0.900
ageCentrd20 -0.233  0.009  0.049  0.026  0.043
```

```
MuMIn::r.squaredGLMM(pointsPerSeason_positionAgeRandomLinearSlopes)
```

```
            R2m       R2c
[1,] 0.09797726 0.5803186
```

```
emmeans::emmeans(pointsPerSeason_positionAgeRandomLinearSlopes, "positionFactor")
```

```
 positionFactor emmean   SE   df lower.CL upper.CL
 FB               12.2 8.46 1315    -4.43     28.8
 QB              100.6 4.21 1367    92.38    108.9
 RB               58.6 2.97 1567    52.81     64.5
 TE               28.1 3.23 1481    21.74     34.4
 WR               45.9 2.41 1588    41.21     50.7

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```
emmeans::emmeans(pointsPerSeason_positionAgeRandomLinearSlopes, "ageCentered20")
```

```
 ageCentered20 emmean   SE   df lower.CL upper.CL
          6.19   49.1 2.15 1309     44.9     53.3

Results are averaged over the levels of: positionFactor
Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```
performance::icc(pointsPerSeason_positionAgeRandomLinearSlopes)
```

```
# Intraclass Correlation Coefficient

    Adjusted ICC: 0.535
  Unadjusted ICC: 0.482
```

```
AIC(pointsPerSeason_positionAgeRandomLinearSlopes)
```

```
[1] 76022.62
```

A plot of the model-implied trajectories of fantasy points by age and position from the mixed model with random intercepts and random linear slopes is in Figure 12.21.

```
pointsPerSeason_positionAge_newData$fantasyPoints_randomLinearSlopes <- predict(
  object = pointsPerSeason_positionAgeRandomLinearSlopes,
  newdata = pointsPerSeason_positionAge_newData,
  re.form = NA
)

ggplot2::ggplot(
  data = pointsPerSeason_positionAge_newData,
  mapping = aes(
    x = age,
    y = fantasyPoints_randomLinearSlopes,
    color = positionFactor
  )
) +
  geom_line(linewidth = 2) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position",
    subtitle = "Mixed Model with Random Intercepts and Random Linear Slopes",
    color = "Position"
  ) +
  theme_classic()
```

Figure 12.21 Plot of Model-Implied Trajectories of Fantasy Points by Age and Position in Mixed Model With Random Intercepts and Random Linear Slopes.

A plot of individuals' model-implied trajectories of fantasy points by age and position from the mixed model with random intercepts and random linear slopes is in Figure 12.21.

```
player_stats_seasonal_offense_subsetCC$fantasyPoints_randomLinearSlopes <- predict(
  object = pointsPerSeason_positionAgeRandomLinearSlopes,
  newdata = player_stats_seasonal_offense_subsetCC
)

plot_individualFantasyPointsRandomLinearSlopes <- ggplot(
  data = player_stats_seasonal_offense_subsetCC %>%
    mutate(
      age = round(age, 2),
      fantasyPoints_randomLinearSlopes = round(fantasyPoints_randomLinearSlopes, 2)
    ),
  mapping = aes(
    x = age,
    y = fantasyPoints_randomLinearSlopes,
    color = positionFactor,
    group = player_id)) +
  geom_line(
    aes(
```

```
      x = age,
      y = fantasyPoints_randomLinearSlopes,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    linewidth = 0.5) +
  geom_line(
    mapping = aes(
      x = age,
      y = fantasyPoints_randomLinearSlopes,
      color = positionFactor
    ),
    data = pointsPerSeason_positionAge_newData %>%
      mutate(
        age = round(age, 2),
        fantasyPoints_randomLinearSlopes = round(fantasyPoints_randomLinearSlopes, 2)
        ),
    inherit.aes = FALSE,
    se = TRUE,
    linewidth = 2
  ) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position:\nModel With Random Intercepts and
    color = "Position"
  ) +
  theme_classic()

ggplotly(
  plot_individualFantasyPointsRandomLinearSlopes,
  tooltip = c("age","fantasyPoints_randomLinearSlopes","text","label")
)
```

Figure 12.22 Plot of Individuals' Implied Trajectories of Fantasy Points by Age and Position, from a Mixed Model With Random Intercepts and Random Linear Slopes. Overlaid with the Model-Implied Trajectory by Position.

12.3.4.3.2 Quadratic Models

12.3.4.3.2.1 Random Intercepts, Random Linear Slopes, Fixed Quadratic Slopes

```r
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes <- lmerTest::lmer(
  fantasy_points ~ positionFactor + ageCentered20 + ageCentered20Quadratic + (1 + ageCentered20 | p
  data = player_stats_seasonal_offense_subset,
  REML = FALSE,
  control = lmerControl(optimizer = "bobyqa")
)

summary(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes)
```

```
Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
  method [lmerModLmerTest]
Formula:
fantasy_points ~ positionFactor + ageCentered20 + ageCentered20Quadratic +
    (1 + ageCentered20 | player_idFactor)
   Data: player_stats_seasonal_offense_subset
Control: lmerControl(optimizer = "bobyqa")

     AIC      BIC   logLik deviance df.resid
 75671.2  75746.4 -37824.6  75649.2     6891

Scaled residuals:
    Min      1Q  Median      3Q     Max
-6.6664 -0.4409 -0.1141  0.3746  4.8142

Random effects:
 Groups          Name          Variance Std.Dev. Corr
 player_idFactor (Intercept)   3856.17  62.098
                 ageCentered20   60.18   7.757   -0.59
 Residual                      2131.00  46.163
Number of obs: 6902, groups:  player_idFactor, 1564

Fixed effects:
                       Estimate Std. Error         df t value Pr(>|t|)
(Intercept)           -21.33523    9.13429 1922.88852  -2.336   0.0196 *
positionFactorQB       88.94900    9.56840 1540.04118   9.296  < 2e-16 ***
positionFactorRB       51.27783    9.07208 1553.82944   5.652 1.88e-08 ***
positionFactorTE       17.26258    9.16457 1543.97093   1.884   0.0598 .
positionFactorWR       37.46611    8.90325 1545.34859   4.208 2.72e-05 ***
ageCentered20          13.54660    0.86539 4903.29411  15.654  < 2e-16 ***
ageCentered20Quadratic  -1.21191    0.05859 3998.41287 -20.683  < 2e-16 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) pstFQB pstFRB pstFTE pstFWR agCn20
postnFctrQB -0.844
postnFctrRB -0.904  0.847
postnFctrTE -0.886  0.838  0.885
postnFctrWR -0.918  0.863  0.912  0.902
ageCentrd20 -0.331  0.004  0.042  0.019  0.036
agCntrd20Qd  0.250  0.004 -0.023 -0.009 -0.020 -0.891
```

```
MuMIn::r.squaredGLMM(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes)
```

```
          R2m       R2c
[1,] 0.163053 0.6703937
```

```
emmeans::emmeans(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes, "positionFactor")
```

```
 positionFactor emmean   SE   df lower.CL upper.CL
 FB               3.33 8.61 1379    -13.6     20.2
 QB              92.28 4.30 1455     83.9    100.7
 RB              54.61 3.02 1627     48.7     60.5
 TE              20.60 3.29 1546     14.1     27.0
 WR              40.80 2.46 1665     36.0     45.6

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```
emmeans::emmeans(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes, "ageCentered20")
```

```
 ageCentered20 emmean   SE   df lower.CL upper.CL
          6.19   42.3 2.23 1385     37.9     46.7

Results are averaged over the levels of: positionFactor
Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```
emmeans::emmeans(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes, "ageCentered20Quadra
```

```
 ageCentered20Quadratic emmean   SE   df lower.CL upper.CL
                   48.8   42.3 2.23 1385     37.9     46.7
```

```
Results are averaged over the levels of: positionFactor
Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
performance::icc(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes)
```

```
# Intraclass Correlation Coefficient

    Adjusted ICC: 0.606
  Unadjusted ICC: 0.507
```

```r
AIC(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes)
```

```
[1] 75671.21
```

A plot of the model-implied trajectories of fantasy points by age and position from the mixed model with random intercepts, random linear slopes, and fixed quadratic slopes is in Figure 12.23.

```r
pointsPerSeason_positionAge_newData$fantasyPoints_randomLinearFixedQuadraticSlopes <- predict(
  object = pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes,
  newdata = pointsPerSeason_positionAge_newData,
  re.form = NA
)

ggplot2::ggplot(
  data = pointsPerSeason_positionAge_newData,
  mapping = aes(
    x = age,
    y = fantasyPoints_randomLinearFixedQuadraticSlopes,
    color = positionFactor
  )
) +
  geom_line(linewidth = 2) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position",
    subtitle = "Mixed Model with Random Intercepts, Random Linear Slopes, and Fixed Quadratic Slop
  ) +
  theme_classic()
```

Figure 12.23 Plot of Model-Implied Trajectories of Fantasy Points by Age and Position in Mixed Model With Random Intercepts, Random Linear Slopes, and Fixed Quadratic Slopes.

A plot of individuals' model-implied trajectories of fantasy points by age and position from the mixed model with random intercepts, random linear slopes, and fixed quadratic slopes is in Figure 12.24.

```
player_stats_seasonal_offense_subsetCC$fantasyPoints_randomLinearFixedQuadraticSlopes <- predict(
  object = pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes,
  newdata = player_stats_seasonal_offense_subsetCC
)

plot_individualFantasyPointsRandomLinearFixedQuadracticSlopes <- ggplot(
  data = player_stats_seasonal_offense_subsetCC %>%
    mutate(
      age = round(age, 2),
      fantasyPoints_randomLinearFixedQuadraticSlopes = round(fantasyPoints_randomLinearFixedQuadra
    ),
  mapping = aes(
    x = age,
    y = fantasyPoints_randomLinearFixedQuadraticSlopes,
    color = positionFactor,
    group = player_id)) +
  geom_line(
    aes(
```

```
      x = age,
      y = fantasyPoints_randomLinearFixedQuadraticSlopes,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    se = FALSE,
    linewidth = 0.5) +
  geom_line(
    mapping = aes(
      x = age,
      y = fantasyPoints_randomLinearFixedQuadraticSlopes,
      color = positionFactor
    ),
    data = pointsPerSeason_positionAge_newData %>%
      mutate(
        age = round(age, 2),
        fantasyPoints_randomLinearFixedQuadraticSlopes = round(fantasyPoints_randomLinearFixedQuad
        ),
    inherit.aes = FALSE,
    se = TRUE,
    linewidth = 2
  ) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position:\nModel With Random Intercepts, Ra
    color = "Position"
  ) +
  theme_classic()

ggplotly(
  plot_individualFantasyPointsRandomLinearFixedQuadracticSlopes,
  tooltip = c("age","fantasyPoints_randomLinearFixedQuadraticSlopes","text","label")
)
```

Figure 12.24 Plot of Individuals' Implied Trajectories of Fantasy Points by Age and Position, from a Mixed Model With Random Intercepts, Random Linear Slopes, and Fixed Quadratic Slopes. Overlaid with the Model-Implied Trajectory by Position.

12.3.4.3.2.2 Random Intercepts, Random Linear Slopes, Random Quadratic Slopes

```
pointsPerSeason_positionAgeRandomLinearRandomQuadraticSlopes <- lmerTest::lmer(
  fantasy_points ~ positionFactor + ageCentered20 + ageCentered20Quadratic + (1 + ageCentered20 +
  data = player_stats_seasonal_offense_subset,
  REML = FALSE,
  control = lmerControl(optimizer = "bobyqa")
)
```

12.3.4.3.2.3 Random Intercepts, Random Linear Slopes, Fixed Quadratic Slopes in Interaction With Position

```
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction <- lmerTest::lmer(
  fantasy_points ~ positionFactor + ageCentered20 + ageCentered20Quadratic + positionFactor:ageCen
  data = player_stats_seasonal_offense_subset,
  REML = FALSE,
  control = lmerControl(optimizer = "bobyqa")
)

summary(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction)
```

```
Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
  method [lmerModLmerTest]
Formula:
fantasy_points ~ positionFactor + ageCentered20 + ageCentered20Quadratic +
   positionFactor:ageCentered20 + positionFactor:ageCentered20Quadratic +
     (1 + ageCentered20 | player_idFactor)
   Data: player_stats_seasonal_offense_subset
Control: lmerControl(optimizer = "bobyqa")

     AIC      BIC   logLik deviance df.resid
 75629.3  75759.2 -37795.6  75591.3     6883

Scaled residuals:
    Min      1Q  Median      3Q     Max
-6.8083 -0.4453 -0.1119  0.3779  4.7783

Random effects:
 Groups          Name          Variance Std.Dev. Corr
```

```
 player_idFactor (Intercept)   3707.88  60.892
                 ageCentered20   56.19   7.496   -0.56
 Residual                      2110.90  45.945
Number of obs: 6902, groups:  player_idFactor, 1564

Fixed effects:
                                    Estimate Std. Error       df t value
(Intercept)                        -23.2382   26.3480 4846.1312  -0.882
positionFactorQB                    70.1298   27.5817 4655.3793   2.543
positionFactorRB                    62.5694   27.1949 4772.1075   2.301
positionFactorTE                    23.3008   27.4084 4771.1591   0.850
positionFactorWR                    34.8154   26.9226 4800.6693   1.293
ageCentered20                       11.8269    7.3841 5202.9022   1.602
ageCentered20Quadratic              -0.8741    0.5056 4717.2629  -1.729
positionFactorQB:ageCentered20       6.4166    7.5876 5170.7168   0.846
positionFactorRB:ageCentered20       0.9003    7.6876 5192.1585   0.117
positionFactorTE:ageCentered20      -1.5181    7.6651 5200.0684  -0.198
positionFactorWR:ageCentered20       3.9903    7.5715 5207.0642   0.527
positionFactorQB:ageCentered20Quadratic  -0.4652    0.5134 4735.7085  -0.906
positionFactorRB:ageCentered20Quadratic  -0.5888    0.5330 4626.1053  -1.105
positionFactorTE:ageCentered20Quadratic   0.0932    0.5247 4682.1399   0.178
positionFactorWR:ageCentered20Quadratic  -0.5803    0.5208 4685.6319  -1.114
                                      Pr(>|t|)
(Intercept)                             0.3778
positionFactorQB                        0.0110 *
positionFactorRB                        0.0214 *
positionFactorTE                        0.3953
positionFactorWR                        0.1960
ageCentered20                           0.1093
ageCentered20Quadratic                  0.0839 .
positionFactorQB:ageCentered20          0.3978
positionFactorRB:ageCentered20          0.9068
positionFactorTE:ageCentered20          0.8430
positionFactorWR:ageCentered20          0.5982
positionFactorQB:ageCentered20Quadratic   0.3649
positionFactorRB:ageCentered20Quadratic   0.2694
positionFactorTE:ageCentered20Quadratic   0.8590
positionFactorWR:ageCentered20Quadratic   0.2652
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
MuMIn::r.squaredGLMM(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction)
```

```
           R2m       R2c
[1,] 0.1559839 0.6709442
```

```
emmeans::emmeans(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction, "position
```

```
 positionFactor emmean   SE   df lower.CL upper.CL
 FB               7.28 9.26 1527    -10.9     25.4
 QB              94.41 4.41 1273     85.8    103.1
 RB              46.67 3.41 1544     40.0     53.4
 TE              25.74 3.50 1426     18.9     32.6
 WR              38.46 2.67 1498     33.2     43.7

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```
emmeans::emmeans(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction, "ageCente
```

```
 ageCentered20 emmean   SE   df lower.CL upper.CL
          6.19   42.5 2.33 1487     37.9     47.1

Results are averaged over the levels of: positionFactor
Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```
emmeans::emmeans(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes, "ageCentered20Quadra
```

```
 ageCentered20Quadratic emmean   SE   df lower.CL upper.CL
                   48.8   42.3 2.23 1385     37.9     46.7

Results are averaged over the levels of: positionFactor
Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```
performance::icc(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction)
```

```
# Intraclass Correlation Coefficient

    Adjusted ICC: 0.610
  Unadjusted ICC: 0.515
```

```
AIC(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction)
```

```
[1] 75629.27
```

A plot of the model-implied trajectories of fantasy points by age and position from the mixed model with random intercepts, random linear slopes, and fixed quadratic slopes in interaction with position is in Figure 12.25.

```
pointsPerSeason_positionAge_newData$fantasyPoints_randomLinearFixedQuadraticSlopesInteraction <- p
  object = pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction,
  newdata = pointsPerSeason_positionAge_newData,
  re.form = NA
)

ggplot2::ggplot(
  data = pointsPerSeason_positionAge_newData,
  mapping = aes(
    x = age,
    y = fantasyPoints_randomLinearFixedQuadraticSlopesInteraction,
    color = positionFactor
  )
) +
  geom_line(linewidth = 2) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position",
    subtitle = "Mixed Model with Random Intercepts, Random Linear Slopes, and Fixed Quadratic Slop
  ) +
  theme_classic()
```

Figure 12.25 Plot of Model-Implied Trajectories of Fantasy Points by Age and Position in Mixed Model With Random Intercepts, Random Linear Slopes, and Fixed Quadratic Slopes in Interaction With Position.

A plot of individuals' model-implied trajectories of fantasy points by age and position from the mixed model with random intercepts, random linear slopes, and fixed quadratic slopes in interaction with position is in Figure 12.26.

```
player_stats_seasonal_offense_subsetCC$fantasyPoints_randomLinearFixedQuadraticSlopesInteraction <-
  object = pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction,
  newdata = player_stats_seasonal_offense_subsetCC
)

plot_individualFantasyPointsRandomLinearFixedQuadraticSlopesInteraction <- ggplot(
  data = player_stats_seasonal_offense_subsetCC %>%
    mutate(
      age = round(age, 2),
      fantasyPoints_randomLinearFixedQuadraticSlopesInteraction = round(fantasyPoints_randomLinear
    ),
  mapping = aes(
    x = age,
    y = fantasyPoints_randomLinearFixedQuadraticSlopesInteraction,
    color = positionFactor,
    group = player_id)) +
  geom_line(
    aes(
```

```
      x = age,
      y = fantasyPoints_randomLinearFixedQuadraticSlopesInteraction,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    linewidth = 0.5) +
  geom_line(
    mapping = aes(
      x = age,
      y = fantasyPoints_randomLinearFixedQuadraticSlopesInteraction,
      color = positionFactor
    ),
    data = pointsPerSeason_positionAge_newData %>%
      mutate(
        age = round(age, 2),
        fantasyPoints_randomLinearFixedQuadraticSlopesInteraction = round(fantasyPoints_randomLine
        ),
    inherit.aes = FALSE,
    se = TRUE,
    linewidth = 2
  ) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position:\nModel With Random Intercepts, Ra
    color = "Position"
  ) +
  theme_classic()

ggplotly(
  plot_individualFantasyPointsRandomLinearFixedQuadraticSlopesInteraction,
  tooltip = c("age","fantasyPoints_randomLinearFixedQuadraticSlopesInteraction","text","label")
)
```

Figure 12.26 Plot of Individuals' Implied Trajectories of Fantasy Points by Age and Position, from a Mixed Model With Random Intercepts, Random Linear Slopes, and Fixed Quadratic Slopes in Interaction With Position. Overlaid with the Model-Implied Trajectory by Position.

12.3.4.3.2.4 Adding Fixed-Effect Predictor of Experience

```r
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperience <- lmerTest::lmer
  fantasy_points ~ positionFactor + ageCentered20 + ageCentered20Quadratic + positionFactor:ageCen
  data = player_stats_seasonal_offense_subset,
  REML = FALSE,
  control = lmerControl(optimizer = "bobyqa")
)

summary(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperience)
```

```
Linear mixed model fit by maximum likelihood . t-tests use Satterthwaite's
  method [lmerModLmerTest]
Formula:
fantasy_points ~ positionFactor + ageCentered20 + ageCentered20Quadratic +
   positionFactor:ageCentered20 + positionFactor:ageCentered20Quadratic +
    years_of_experience + (1 + ageCentered20 | player_idFactor)
   Data: player_stats_seasonal_offense_subset
Control: lmerControl(optimizer = "bobyqa")

     AIC      BIC   logLik deviance df.resid
 75544.8  75681.6 -37752.4  75504.8     6878

Scaled residuals:
   Min     1Q Median     3Q    Max
-6.776 -0.441 -0.108  0.378  4.748

Random effects:
 Groups          Name          Variance Std.Dev. Corr
 player_idFactor (Intercept)   3686.7   60.718
                 ageCentered20   54.9    7.409   -0.58
 Residual                      2115.6   45.995
Number of obs: 6898, groups:  player_idFactor, 1561

Fixed effects:
                                        Estimate Std. Error         df
(Intercept)                            -14.50822   26.37172 4868.85153
positionFactorQB                        66.97940   27.57429 4672.43127
positionFactorRB                        60.25007   27.18918 4787.83437
positionFactorTE                        22.55466   27.39867 4785.40457
positionFactorWR                        32.47573   26.91538 4816.29115
ageCentered20                            6.56788    7.42196 5300.00305
ageCentered20Quadratic                  -0.90467    0.50532 4700.19676
```

```
years_of_experience                      6.54473    0.98840 2355.10318
positionFactorQB:ageCentered20           6.29582    7.58289 5171.33712
positionFactorRB:ageCentered20           1.07928    7.68326 5191.55716
positionFactorTE:ageCentered20          -1.64723    7.66075 5199.16675
positionFactorWR:ageCentered20           4.06073    7.56721 5206.56507
positionFactorQB:ageCentered20Quadratic  -0.44894    0.51313 4719.21422
positionFactorRB:ageCentered20Quadratic  -0.61009    0.53273 4610.56779
positionFactorTE:ageCentered20Quadratic   0.09582    0.52438 4665.21647
positionFactorWR:ageCentered20Quadratic  -0.58780    0.52051 4670.05797
                                        t value Pr(>|t|)
(Intercept)                              -0.550   0.5822
positionFactorQB                          2.429   0.0152 *
positionFactorRB                          2.216   0.0267 *
positionFactorTE                          0.823   0.4104
positionFactorWR                          1.207   0.2277
ageCentered20                             0.885   0.3762
ageCentered20Quadratic                   -1.790   0.0735 .
years_of_experience                       6.622 4.39e-11 ***
positionFactorQB:ageCentered20            0.830   0.4064
positionFactorRB:ageCentered20            0.140   0.8883
positionFactorTE:ageCentered20           -0.215   0.8298
positionFactorWR:ageCentered20            0.537   0.5916
positionFactorQB:ageCentered20Quadratic  -0.875   0.3817
positionFactorRB:ageCentered20Quadratic  -1.145   0.2522
positionFactorTE:ageCentered20Quadratic   0.183   0.8550
positionFactorWR:ageCentered20Quadratic  -1.129   0.2588
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
MuMIn::r.squaredGLMM(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperie
```

```
         R2m       R2c
[1,] 0.16495 0.6649985
```

```
emmeans::emmeans(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperience,
```

```
 positionFactor emmean   SE   df lower.CL upper.CL
 FB               10.9 9.11 1536    -7.01     28.7
 QB               94.9 4.33 1269    86.40    103.4
 RB               48.0 3.36 1533    41.41     54.6
 TE               27.9 3.44 1425    21.14     34.7
 WR               39.8 2.63 1490    34.61     44.9

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
emmeans::emmeans(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperience,
```

```
 ageCentered20 emmean  SE   df lower.CL upper.CL
          6.19   44.3 2.3 1491     39.8     48.8

Results are averaged over the levels of: positionFactor
Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
emmeans::emmeans(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperience,
```

```
 ageCentered20Quadratic emmean  SE   df lower.CL upper.CL
                   48.9   44.3 2.3 1491     39.8     48.8

Results are averaged over the levels of: positionFactor
Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
emmeans::emmeans(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperience,
```

```
 years_of_experience emmean  SE   df lower.CL upper.CL
                4.42   44.3 2.3 1491     39.8     48.8

Results are averaged over the levels of: positionFactor
Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
performance::icc(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperience)
```

```
# Intraclass Correlation Coefficient

    Adjusted ICC: 0.599
  Unadjusted ICC: 0.500
```

```r
AIC(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperience)
```

```
[1] 75544.81
```

12.3.4.3.3 Compare Models

After fitting several models, we now must compare their fit to determine which model fits "best" while also considering parsimony. Parsimonious models are more likely to be true and more likely to generalize to other samples, because more complex models are more likely to overfit the data. Thus, more complex models will almost always fit better than simpler models. Thus, we are not just interested in whether a more complex model fits better than the simpler model; we also care about whether the more complex model fits *significantly* better than the simpler model given its additional complexity. For evaluating and comparing models, we examine the likelihood ratio test, the Akaike Information Criterion (AIC), the corrected AIC (AICc), the Bayesian Information Criterion (BIC), R^2, deviance, and log likelihood.

The BIC penalizes model complexity more than the AIC does. The BIC is preferable when there is a "true" model, and one intends to identify the true model. The AIC is preferable when we are concerned more about predictive accuracy and when overfitting is less of a concern. Because we are more concerned about predictive accuracy and we do not believe one of these models is the "true" model per se of age-related changes in fantasy performance, we will give more weight to AIC than BIC.

Below, we specify various groups of models for the model fit comparisons:

```
lmVsMixedModel <- list(
  "nullModel" = pointsPerSeason_nullModel,
  "randomIntercepts" = pointsPerSeason_randomIntercepts
)

lmAndMixedModels <- list(
  "nullModel" = pointsPerSeason_nullModel,
  "linearRegression" = pointsPerSeason_linearRegression,
  "quadraticRegression" = pointsPerSeason_quadraticRegression,
  "randomIntercepts" = pointsPerSeason_randomIntercepts,
  "position" = pointsPerSeason_position,
  "fixedLinear" = pointsPerSeason_positionAgeFixedLinearSlopes,
  "randomLinear" = pointsPerSeason_positionAgeRandomLinearSlopes,
  "randomLinearFixedQuadratic" = pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes,
  "randomLinearFixedQuadraticInteraction" = pointsPerSeason_positionAgeRandomLinearFixedQuadraticS
)

mixedModels <- list(
  "randomIntercepts" = pointsPerSeason_randomIntercepts,
  "position" = pointsPerSeason_position,
  "fixedLinear" = pointsPerSeason_positionAgeFixedLinearSlopes,
```

```
  "randomLinear" = pointsPerSeason_positionAgeRandomLinearSlopes,
  "randomLinearFixedQuadratic" = pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes,
  "randomLinearFixedQuadraticInteraction" = pointsPerSeason_positionAgeRandomLinearFixedQuadraticS
)

mixedModels1 <- list(
  "randomIntercepts" = pointsPerSeason_randomIntercepts,
  "position" = pointsPerSeason_position
)

mixedModels2 <- list(
  "fixedLinear" = pointsPerSeason_positionAgeFixedLinearSlopes,
  "randomLinear" = pointsPerSeason_positionAgeRandomLinearSlopes,
  "randomLinearFixedQuadratic" = pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes,
  "randomLinearFixedQuadraticInteraction" = pointsPerSeason_positionAgeRandomLinearFixedQuadraticS
)
```

12.3.4.3.3.1 Likelihood Ratio Test

```
anova(
  pointsPerSeason_randomIntercepts,
  pointsPerSeason_position
)
```

```
Data: player_stats_seasonal_offense_subset
Models:
pointsPerSeason_randomIntercepts: fantasy_points ~ 1 + (1 | player_idFactor)
pointsPerSeason_position: fantasy_points ~ positionFactor + (1 | player_idFactor)
                            npar    AIC    BIC logLik deviance  Chisq Df
pointsPerSeason_randomIntercepts    3 148044 148067 -74019   148038
pointsPerSeason_position        7 147766 147818 -73876  147752 286.37  4
                                 Pr(>Chisq)
pointsPerSeason_randomIntercepts
pointsPerSeason_position          < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(
  pointsPerSeason_positionAgeFixedLinearSlopes,
  pointsPerSeason_positionAgeRandomLinearSlopes,
  pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes,
```

```
  pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction
)
```

```
Data: player_stats_seasonal_offense_subset
Models:
pointsPerSeason_positionAgeFixedLinearSlopes: fantasy_points ~ positionFactor + ageCentered20 + (1 |
pointsPerSeason_positionAgeRandomLinearSlopes: fantasy_points ~ positionFactor + ageCentered20 + (1 +
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes: fantasy_points ~ positionFactor + ageCe
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction: fantasy_points ~ positionFa
                                                                   npar
pointsPerSeason_positionAgeFixedLinearSlopes                          8
pointsPerSeason_positionAgeRandomLinearSlopes                        10
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes          11
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction  19
                                                                    AIC
pointsPerSeason_positionAgeFixedLinearSlopes                      76252
pointsPerSeason_positionAgeRandomLinearSlopes                     76023
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes        75671
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction 75629
                                                                    BIC
pointsPerSeason_positionAgeFixedLinearSlopes                      76307
pointsPerSeason_positionAgeRandomLinearSlopes                     76091
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes        75746
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction 75759
                                                                 logLik
pointsPerSeason_positionAgeFixedLinearSlopes                    -38118
pointsPerSeason_positionAgeRandomLinearSlopes                   -38001
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes        -37825
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction -37796
                                                               deviance
pointsPerSeason_positionAgeFixedLinearSlopes                      76236
pointsPerSeason_positionAgeRandomLinearSlopes                     76003
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes          75649
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction   75591
                                                                  Chisq
pointsPerSeason_positionAgeFixedLinearSlopes
pointsPerSeason_positionAgeRandomLinearSlopes                   233.669
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes        353.408
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction  57.943
                                                                     Df
pointsPerSeason_positionAgeFixedLinearSlopes
pointsPerSeason_positionAgeRandomLinearSlopes                         2
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes           1
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction  8
```

```
                                                                Pr(>Chisq)
pointsPerSeason_positionAgeFixedLinearSlopes
pointsPerSeason_positionAgeRandomLinearSlopes                    < 2.2e-16
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes        < 2.2e-16
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction  1.179e-09

pointsPerSeason_positionAgeFixedLinearSlopes
pointsPerSeason_positionAgeRandomLinearSlopes                       ***
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes        ***
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

12.3.4.3.3.2 Akaike Information Criterion (AIC)

```r
AIC(
  pointsPerSeason_nullModel,
  pointsPerSeason_linearRegression,
  pointsPerSeason_quadraticRegression,
  pointsPerSeason_randomIntercepts,
  pointsPerSeason_positionAgeFixedLinearSlopes,
  pointsPerSeason_positionAgeRandomLinearSlopes,
  pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes,
  pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction
  )
```

```
                                                                df
pointsPerSeason_nullModel                                        2
pointsPerSeason_linearRegression                                11
pointsPerSeason_quadraticRegression                             16
pointsPerSeason_randomIntercepts                                 3
pointsPerSeason_positionAgeFixedLinearSlopes                     8
pointsPerSeason_positionAgeRandomLinearSlopes                   10
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes     11
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction 19
                                                               AIC
pointsPerSeason_nullModel                                154719.69
pointsPerSeason_linearRegression                          79199.17
pointsPerSeason_quadraticRegression                       79187.85
pointsPerSeason_randomIntercepts                         148044.03
pointsPerSeason_positionAgeFixedLinearSlopes              76252.29
pointsPerSeason_positionAgeRandomLinearSlopes             76022.62
```

```
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes        75671.21
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction 75629.27
```

```
bbmle::AICtab(lmAndMixedModels)
```

```
                                    dAIC    df
randomLinearFixedQuadraticInteraction     0.0 19
randomLinearFixedQuadratic               41.9 11
randomLinear                            393.4 10
fixedLinear                             623.0 8
quadraticRegression                    3558.6 16
linearRegression                       3569.9 11
position                              72136.4 7
randomIntercepts                      72414.8 3
nullModel                             79090.4 2
```

12.3.4.3.3.3 Corrected Akaike Information Criterion (AICc)

```
#AICcmodavg::aictab(lmVsMixedModel) # throws error (can't mix lm with lmer)
bbmle::AICctab(lmVsMixedModel)
```

```
                 dAICc  df
randomIntercepts    0.0 3
nullModel        6675.7 2
```

```
AICcmodavg::aictab(mixedModels) # throws error (can't mix lm with lmer)
```

```
Model selection based on AICc:

                                  K      AICc Delta_AICc AICcWt Cum.Wt
randomLinearFixedQuadraticInteraction 19  75629.38      0.00     1     1
randomLinearFixedQuadratic          11  75671.25     41.87     0     1
randomLinear                       10  76022.65    393.27     0     1
fixedLinear                         8  76252.31    622.93     0     1
position                            7 147765.67   72136.29     0     1
randomIntercepts                    3 148044.03   72414.65     0     1
                                            LL
randomLinearFixedQuadraticInteraction -37795.64
randomLinearFixedQuadratic            -37824.61
randomLinear                          -38001.31
```

```
fixedLinear                          -38118.14
position                             -73875.83
randomIntercepts                     -74019.01
```

```
#bbmle::AICctab(mixedModels) # throws error (different numbers of observations)

AICcmodavg::aictab(mixedModels1)
```

```
Model selection based on AICc:

                 K     AICc Delta_AICc AICcWt Cum.Wt        LL
position         7 147765.7       0.00      1      1 -73875.83
randomIntercepts 3 148044.0     278.36      0      1 -74019.01
```

```
bbmle::AICctab(mixedModels1)
```

```
                 dAICc df
position           0.0 7
randomIntercepts 278.4 3
```

```
AICcmodavg::aictab(mixedModels2)
```

```
Model selection based on AICc:

                                   K     AICc Delta_AICc AICcWt Cum.Wt
randomLinearFixedQuadraticInteraction 19 75629.38      0.00     1     1
randomLinearFixedQuadratic           11 75671.25     41.87     0     1
randomLinear                         10 76022.65    393.27     0     1
fixedLinear                           8 76252.31    622.93     0     1
                                            LL
randomLinearFixedQuadraticInteraction -37795.64
randomLinearFixedQuadratic            -37824.61
randomLinear                          -38001.31
fixedLinear                           -38118.14
```

```
bbmle::AICctab(mixedModels2)
```

```
                                   dAICc df
randomLinearFixedQuadraticInteraction   0.0 19
randomLinearFixedQuadratic             41.9 11
randomLinear                          393.3 10
fixedLinear                           622.9 8
```

```
MuMIn::AICc(
  pointsPerSeason_nullModel,
  pointsPerSeason_linearRegression,
  pointsPerSeason_quadraticRegression,
  pointsPerSeason_randomIntercepts,
  pointsPerSeason_positionAgeFixedLinearSlopes,
  pointsPerSeason_positionAgeRandomLinearSlopes,
  pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes,
  pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction
)
```

```
                                                                  df
pointsPerSeason_nullModel                                          2
pointsPerSeason_linearRegression                                  11
pointsPerSeason_quadraticRegression                               16
pointsPerSeason_randomIntercepts                                   3
pointsPerSeason_positionAgeFixedLinearSlopes                       8
pointsPerSeason_positionAgeRandomLinearSlopes                     10
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes       11
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction 19
                                                                 AICc
pointsPerSeason_nullModel                                   154719.69
pointsPerSeason_linearRegression                             79199.21
pointsPerSeason_quadraticRegression                          79187.93
pointsPerSeason_randomIntercepts                            148044.03
pointsPerSeason_positionAgeFixedLinearSlopes                 76252.31
pointsPerSeason_positionAgeRandomLinearSlopes                76022.65
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes        75671.25
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction  75629.38
```

12.3.4.3.3.4 Bayesian Information Criterion (BIC)

```
BIC(
  pointsPerSeason_nullModel,
  pointsPerSeason_linearRegression,
  pointsPerSeason_quadraticRegression,
  pointsPerSeason_randomIntercepts,
  pointsPerSeason_positionAgeFixedLinearSlopes,
  pointsPerSeason_positionAgeRandomLinearSlopes,
  pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes,
  pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction
  )
```

```
                                                                  df
pointsPerSeason_nullModel                                          2
pointsPerSeason_linearRegression                                  11
pointsPerSeason_quadraticRegression                               16
pointsPerSeason_randomIntercepts                                   3
pointsPerSeason_positionAgeFixedLinearSlopes                       8
pointsPerSeason_positionAgeRandomLinearSlopes                     10
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes       11
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction 19
                                                                 BIC
pointsPerSeason_nullModel                                   154734.72
pointsPerSeason_linearRegression                             79274.41
pointsPerSeason_quadraticRegression                          79297.28
pointsPerSeason_randomIntercepts                            148066.57
pointsPerSeason_positionAgeFixedLinearSlopes                 76307.01
pointsPerSeason_positionAgeRandomLinearSlopes                76091.02
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes       75746.45
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction 75759.22
```

```
#AICcmodavg::bictab(lmAndMixedModels) # throws error (can't mix lm with lmer)
bbmle::BICtab(lmVsMixedModel)
```

```
                 dBIC   df
randomIntercepts    0.0 3
nullModel        6668.2 2
```

```
AICcmodavg::bictab(mixedModels)
```

```
Model selection based on BIC:

                                   K       BIC Delta_BIC BICWt Cum.Wt
randomLinearFixedQuadratic        11  75746.45      0.00     1      1
randomLinearFixedQuadraticInteraction 19  75759.22    12.77     0      1
randomLinear                      10  76091.02    344.57     0      1
fixedLinear                        8  76307.01    560.56     0      1
position                           7 147818.25  72071.80     0      1
randomIntercepts                   3 148066.57  72320.12     0      1
                                          LL
randomLinearFixedQuadratic          -37824.61
randomLinearFixedQuadraticInteraction -37795.64
randomLinear                        -38001.31
fixedLinear                         -38118.14
position                            -73875.83
randomIntercepts                    -74019.01
```

```
#bbmle::AICctab(mixedModels) # throws error (different numbers of observations)

AICcmodavg::bictab(mixedModels1)
```

```
Model selection based on BIC:

                 K      BIC Delta_BIC BICWt Cum.Wt        LL
position         7 147818.2      0.00     1      1 -73875.83
randomIntercepts 3 148066.6    248.32     0      1 -74019.01
```

```
bbmle::BICtab(mixedModels1)
```

```
                 dBIC  df
position           0.0 7
randomIntercepts 248.3 3
```

```
AICcmodavg::bictab(mixedModels2)
```

```
Model selection based on BIC:

                                     K      BIC Delta_BIC BICWt Cum.Wt
randomLinearFixedQuadratic          11 75746.45      0.00     1      1
randomLinearFixedQuadraticInteraction 19 75759.22     12.77     0      1
randomLinear                        10 76091.02    344.57     0      1
fixedLinear                          8 76307.01    560.56     0      1
                                            LL
randomLinearFixedQuadratic           -37824.61
randomLinearFixedQuadraticInteraction -37795.64
randomLinear                         -38001.31
fixedLinear                          -38118.14
```

```
bbmle::BICtab(mixedModels2)
```

```
                                     dBIC  df
randomLinearFixedQuadratic             0.0 11
randomLinearFixedQuadraticInteraction  12.8 19
randomLinear                          344.6 10
fixedLinear                           560.6 8
```

12.3.4.3.3.5 R^2

```
summary(pointsPerSeason_nullModel)$r.squared
```

```
[1] 0
```

```
summary(pointsPerSeason_linearRegression)$r.squared
```

```
[1] 0.1410917
```

```
summary(pointsPerSeason_quadraticRegression)$r.squared
```

```
[1] 0.1437412
```

```
MuMIn::r.squaredGLMM(pointsPerSeason_randomIntercepts)
```

```
     R2m       R2c
[1,]   0 0.4847453
```

```
MuMIn::r.squaredGLMM(pointsPerSeason_positionAgeFixedLinearSlopes)
```

```
            R2m       R2c
[1,] 0.09870768 0.5351623
```

```
MuMIn::r.squaredGLMM(pointsPerSeason_positionAgeRandomLinearSlopes)
```

```
            R2m       R2c
[1,] 0.09797726 0.5803186
```

```
MuMIn::r.squaredGLMM(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes)
```

```
          R2m       R2c
[1,] 0.163053 0.6703937
```

```
MuMIn::r.squaredGLMM(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction)
```

```
           R2m       R2c
[1,] 0.1559839 0.6709442
```

12.3.4.3.3.6 Deviance

```
deviance(pointsPerSeason_nullModel)
```

```
[1] 73167060
```

```
deviance(pointsPerSeason_linearRegression)
```

```
[1] 38776196
```

```
deviance(pointsPerSeason_quadraticRegression)
```

```
[1] 38656580
```

```
deviance(pointsPerSeason_randomIntercepts)
```

```
[1] 148038
```

```
deviance(pointsPerSeason_positionAgeFixedLinearSlopes)
```

```
[1] 76236.29
```

```
deviance(pointsPerSeason_positionAgeRandomLinearSlopes)
```

```
[1] 76002.62
```

```
deviance(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes)
```

```
[1] 75649.21
```

```
deviance(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction)
```

```
[1] 75591.27
```

12.3.4.3.3.7 Log Likelihood

```
logLik(pointsPerSeason_nullModel)
```

```
'log Lik.' -77357.85 (df=2)
```

```
logLik(pointsPerSeason_linearRegression)
```

```
'log Lik.' -39588.59 (df=11)
```

```
logLik(pointsPerSeason_quadraticRegression)
```

```
'log Lik.' -39577.93 (df=16)
```

```
logLik(pointsPerSeason_randomIntercepts)
```

```
'log Lik.' -74019.01 (df=3)
```

```
logLik(pointsPerSeason_positionAgeFixedLinearSlopes)
```

```
'log Lik.' -38118.14 (df=8)
```

```
logLik(pointsPerSeason_positionAgeRandomLinearSlopes)
```

```
'log Lik.' -38001.31 (df=10)
```

```
logLik(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes)
```

```
'log Lik.' -37824.61 (df=11)
```

```
logLik(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction)
```

```
'log Lik.' -37795.64 (df=19)
```

12.3.4.3.4 Generalized Additive Model

```
num_cores <- detectCores()
num_cores
```

```
[1] 4
```

```r
pointsPerSeason_gam <- bam( # using bam() instead of gam() for faster estimation due to large size
  fantasy_points ~ positionFactor + s(ageCentered20, by = positionFactor) + years_of_experience + s
  data = player_stats_seasonal_offense_subset,
  nthreads = num_cores
)

pointsPerSeason_gamSummary <- summary(pointsPerSeason_gam)

pointsPerSeason_gamSummary
```

```
Family: gaussian
Link function: identity

Formula:
fantasy_points ~ positionFactor + s(ageCentered20, by = positionFactor) +
    years_of_experience + s(player_idFactor, ageCentered20, bs = "re")

Parametric coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)          -8.5411     9.9541  -0.858  0.39090
positionFactorQB     87.7757    10.4222   8.422  < 2e-16 ***
positionFactorRB     29.2312    10.0185   2.918  0.00354 **
positionFactorTE     13.3524    10.0285   1.331  0.18310
positionFactorWR     23.2301     9.7602   2.380  0.01734 *
years_of_experience   4.6862     0.8807   5.321 1.07e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                                     edf   Ref.df      F p-value
s(ageCentered20):positionFactorFB   1.793    2.263  2.801  0.0516 .
s(ageCentered20):positionFactorQB   5.228    6.402 36.850  <2e-16 ***
s(ageCentered20):positionFactorRB   4.914    5.920 38.784  <2e-16 ***
s(ageCentered20):positionFactorTE   4.209    5.220 15.268  <2e-16 ***
s(ageCentered20):positionFactorWR   5.404    6.401 42.369  <2e-16 ***
s(player_idFactor,ageCentered20)  968.689 1556.000  4.341  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.576   Deviance explained = 63.7%
fREML =  38169  Scale est. = 2774.6    n = 6898
```

```
pointsPerSeason_gamSummary$r.sq
```

```
[1] 0.5759413
```

```
MuMIn::r.squaredGLMM(pointsPerSeason_gam)
```

```
           R2m       R2c
[1,] 0.5488919 0.5488919
```

```
AIC(pointsPerSeason_gam)
```

```
[1] 75187.43
```

12.3.4.3.4.1 Compare Models

```
linearMixedModelsVsGAM <- list(
  "fixedLinear" = pointsPerSeason_positionAgeFixedLinearSlopes,
  "randomLinear" = pointsPerSeason_positionAgeRandomLinearSlopes,
  "randomLinearFixedQuadratic" = pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes,
  "randomLinearFixedQuadraticInteraction" = pointsPerSeason_positionAgeRandomLinearFixedQuadraticS
  "gam" = pointsPerSeason_gam
)

AIC(
  #pointsPerSeason_nullModel,
  #pointsPerSeason_linearRegression,
  #pointsPerSeason_quadraticRegression,
  #pointsPerSeason_randomIntercepts,
  pointsPerSeason_positionAgeFixedLinearSlopes,
  pointsPerSeason_positionAgeRandomLinearSlopes,
  pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes,
  pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction,
  pointsPerSeason_gam
)
```

```
                                                                  df
pointsPerSeason_positionAgeFixedLinearSlopes                   8.0000
pointsPerSeason_positionAgeRandomLinearSlopes                 10.0000
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes        11.0000
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction 19.0000
```

```
pointsPerSeason_gam                                          999.2739
                                                                  AIC
pointsPerSeason_positionAgeFixedLinearSlopes                 76252.29
pointsPerSeason_positionAgeRandomLinearSlopes                76022.62
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes        75671.21
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction 75629.27
pointsPerSeason_gam                                          75187.43
```

```r
#AICcmodavg::aictab(linearMixedModelsVsGAM) # throws error (can't mix bam with lmer)
#bbmle::AICctab(linearMixedModelsVsGAM) # different numbers of observations

MuMIn::AICc(
  pointsPerSeason_nullModel,
  pointsPerSeason_linearRegression,
  pointsPerSeason_quadraticRegression,
  pointsPerSeason_randomIntercepts,
  pointsPerSeason_positionAgeFixedLinearSlopes,
  pointsPerSeason_positionAgeRandomLinearSlopes,
  pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes,
  pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction,
  pointsPerSeason_gam
)
```

```
                                                                     df
pointsPerSeason_nullModel                                        2.0000
pointsPerSeason_linearRegression                                11.0000
pointsPerSeason_quadraticRegression                             16.0000
pointsPerSeason_randomIntercepts                                 3.0000
pointsPerSeason_positionAgeFixedLinearSlopes                     8.0000
pointsPerSeason_positionAgeRandomLinearSlopes                   10.0000
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes        11.0000
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction 19.0000
pointsPerSeason_gam                                          999.2739
                                                                 AICc
pointsPerSeason_nullModel                                   154719.69
pointsPerSeason_linearRegression                             79199.21
pointsPerSeason_quadraticRegression                          79187.93
pointsPerSeason_randomIntercepts                            148044.03
pointsPerSeason_positionAgeFixedLinearSlopes                 76252.31
pointsPerSeason_positionAgeRandomLinearSlopes                76022.65
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes        75671.25
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction 75629.38
pointsPerSeason_gam                                          75526.39
```

```r
BIC(
  #pointsPerSeason_nullModel,
  #pointsPerSeason_linearRegression,
  #pointsPerSeason_quadraticRegression,
  #pointsPerSeason_randomIntercepts,
  pointsPerSeason_positionAgeFixedLinearSlopes,
  pointsPerSeason_positionAgeRandomLinearSlopes,
  pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes,
  pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction,
  pointsPerSeason_gam
)
```

```
                                                                  df
pointsPerSeason_positionAgeFixedLinearSlopes                   8.0000
pointsPerSeason_positionAgeRandomLinearSlopes                 10.0000
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes        11.0000
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction 19.0000
pointsPerSeason_gam                                          999.2739
                                                                 BIC
pointsPerSeason_positionAgeFixedLinearSlopes                 76307.01
pointsPerSeason_positionAgeRandomLinearSlopes                76091.02
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes        75746.45
pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction 75759.22
pointsPerSeason_gam                                          82021.45
```

```r
#AICcmodavg::bictab(linearMixedModelsVsGAM) # throws error (can't mix bam with lmer)
#bbmle::AICctab(linearMixedModelsVsGAM) # different numbers of observations

summary(pointsPerSeason_nullModel)$r.squared
```

```
[1] 0
```

```r
summary(pointsPerSeason_linearRegression)$r.squared
```

```
[1] 0.1410917
```

```r
summary(pointsPerSeason_quadraticRegression)$r.squared
```

```
[1] 0.1437412
```

```
MuMIn::r.squaredGLMM(pointsPerSeason_randomIntercepts)
```

```
     R2m       R2c
[1,]   0 0.4847453
```

```
MuMIn::r.squaredGLMM(pointsPerSeason_positionAgeFixedLinearSlopes)
```

```
            R2m       R2c
[1,] 0.09870768 0.5351623
```

```
MuMIn::r.squaredGLMM(pointsPerSeason_positionAgeRandomLinearSlopes)
```

```
            R2m       R2c
[1,] 0.09797726 0.5803186
```

```
MuMIn::r.squaredGLMM(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes)
```

```
          R2m       R2c
[1,] 0.163053 0.6703937
```

```
MuMIn::r.squaredGLMM(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction)
```

```
           R2m       R2c
[1,] 0.1559839 0.6709442
```

```
MuMIn::r.squaredGLMM(pointsPerSeason_gam)
```

```
           R2m       R2c
[1,] 0.5488919 0.5488919
```

```
deviance(pointsPerSeason_nullModel)
```

```
[1] 73167060
```

```
deviance(pointsPerSeason_linearRegression)
```

```
[1] 38776196
```

```r
deviance(pointsPerSeason_quadraticRegression)
```

```
[1] 38656580
```

```r
deviance(pointsPerSeason_randomIntercepts)
```

```
[1] 148038
```

```r
deviance(pointsPerSeason_positionAgeFixedLinearSlopes)
```

```
[1] 76236.29
```

```r
deviance(pointsPerSeason_positionAgeRandomLinearSlopes)
```

```
[1] 76002.62
```

```r
deviance(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes)
```

```
[1] 75649.21
```

```r
deviance(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction)
```

```
[1] 75591.27
```

```r
deviance(pointsPerSeason_gam)
```

```
[1] 16375053
```

```r
logLik(pointsPerSeason_nullModel)
```

```
'log Lik.' -77357.85 (df=2)
```

```r
logLik(pointsPerSeason_linearRegression)
```

```
'log Lik.' -39588.59 (df=11)
```

```r
logLik(pointsPerSeason_quadraticRegression)
```

```
'log Lik.' -39577.93 (df=16)
```

```r
logLik(pointsPerSeason_randomIntercepts)
```

```
'log Lik.' -74019.01 (df=3)
```

```r
logLik(pointsPerSeason_positionAgeFixedLinearSlopes)
```

```
'log Lik.' -38118.14 (df=8)
```

```r
logLik(pointsPerSeason_positionAgeRandomLinearSlopes)
```

```
'log Lik.' -38001.31 (df=10)
```

```r
logLik(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopes)
```

```
'log Lik.' -37824.61 (df=11)
```

```r
logLik(pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteraction)
```

```
'log Lik.' -37795.64 (df=19)
```

```r
logLik(pointsPerSeason_gam)
```

```
'log Lik.' -36594.44 (df=999.2739)
```

12.3.4.3.5 Players Who Were (at Least Once) at the Top of the End-of-Season Depth Chart

```r
pointsPerSeasonDepth_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperience <- lmerTest:
  fantasy_points ~ positionFactor + ageCentered20 + ageCentered20Quadratic + positionFactor:ageCen
  data = player_stats_seasonal_offense_subset,
  control = lmerControl(optimizer = "bobyqa")
)

summary(pointsPerSeasonDepth_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperience)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula:
fantasy_points ~ positionFactor + ageCentered20 + ageCentered20Quadratic +
   positionFactor:ageCentered20 + positionFactor:ageCentered20Quadratic +
    years_of_experience + (1 + ageCentered20 | player_idFactor)
   Data: player_stats_seasonal_offense_subset
Control: lmerControl(optimizer = "bobyqa")

REML criterion at convergence: 75480.3

Scaled residuals:
    Min      1Q  Median      3Q     Max
-6.7766 -0.4408 -0.1074  0.3776  4.7433

Random effects:
 Groups          Name          Variance Std.Dev. Corr
 player_idFactor (Intercept)   3717.29  60.970
                 ageCentered20   55.82   7.471   -0.58
 Residual                      2117.24  46.013
Number of obs: 6898, groups:  player_idFactor, 1561

Fixed effects:
                                    Estimate Std. Error       df t value
(Intercept)                        -14.5968   26.4120 4851.1705  -0.553
positionFactorQB                    66.9710   27.6178 4653.8846   2.425
positionFactorRB                    60.2312   27.2312 4770.0039   2.212
positionFactorTE                    22.5893   27.4411 4767.5260   0.823
positionFactorWR                    32.4586   26.9568 4798.5095   1.204
ageCentered20                        6.6008    7.4328 5290.5971   0.888
ageCentered20Quadratic              -0.9070    0.5061 4698.2892  -1.792
years_of_experience                  6.5381    0.9906 2345.9972   6.600
positionFactorQB:ageCentered20        6.3045    7.5941 5161.3335   0.830
positionFactorRB:ageCentered20        1.0953    7.6945 5182.4236   0.142
positionFactorTE:ageCentered20       -1.6561    7.6719 5190.0409  -0.216
positionFactorWR:ageCentered20        4.0754    7.5782 5197.3345   0.538
positionFactorQB:ageCentered20Quadratic  -0.4506    0.5140 4717.5388  -0.877
positionFactorRB:ageCentered20Quadratic  -0.6127    0.5336 4609.5912  -1.148
positionFactorTE:ageCentered20Quadratic   0.0961    0.5252 4664.3414   0.183
positionFactorWR:ageCentered20Quadratic  -0.5901    0.5213 4668.6479  -1.132
                                      Pr(>|t|)
(Intercept)                             0.5805
positionFactorQB                        0.0153 *
positionFactorRB                        0.0270 *
positionFactorTE                        0.4104
```

```
positionFactorWR                          0.2286
ageCentered20                             0.3745
ageCentered20Quadratic                    0.0732 .
years_of_experience                     5.06e-11 ***
positionFactorQB:ageCentered20            0.4065
positionFactorRB:ageCentered20            0.8868
positionFactorTE:ageCentered20            0.8291
positionFactorWR:ageCentered20            0.5908
positionFactorQB:ageCentered20Quadratic   0.3807
positionFactorRB:ageCentered20Quadratic   0.2509
positionFactorTE:ageCentered20Quadratic   0.8548
positionFactorWR:ageCentered20Quadratic   0.2578
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
MuMIn::r.squaredGLMM(pointsPerSeasonDepth_positionAgeRandomLinearFixedQuadraticSlopesInteractionEx
```

```
           R2m       R2c
[1,] 0.1647783 0.6662984
```

```
emmeans::emmeans(pointsPerSeasonDepth_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperi
```

```
 positionFactor emmean   SE   df lower.CL upper.CL
 FB               10.8 9.11 1517    -7.03     28.7
 QB               94.8 4.33 1252    86.34    103.3
 RB               47.9 3.36 1513    41.33     54.5
 TE               27.9 3.44 1408    21.11     34.6
 WR               39.7 2.63 1470    34.54     44.9

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```
emmeans::emmeans(pointsPerSeasonDepth_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperi
```

```
 ageCentered20 emmean  SE   df lower.CL upper.CL
          6.19   44.2 2.3 1472     39.7     48.7

Results are averaged over the levels of: positionFactor
Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```
emmeans::emmeans(pointsPerSeasonDepth_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperi
```

```
 ageCentered20Quadratic emmean  SE   df lower.CL upper.CL
                   48.9   44.2 2.3 1472     39.7     48.7

Results are averaged over the levels of: positionFactor
Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```
emmeans::emmeans(pointsPerSeasonDepth_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperi
```

```
 years_of_experience emmean  SE   df lower.CL upper.CL
                4.42   44.2 2.3 1472     39.7     48.7

Results are averaged over the levels of: positionFactor
Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```
performance::icc(pointsPerSeasonDepth_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperi
```

```
# Intraclass Correlation Coefficient

    Adjusted ICC: 0.600
  Unadjusted ICC: 0.502
```

```
AIC(pointsPerSeasonDepth_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperience)
```

```
[1] 75520.29
```

```
pointsPerSeasonDepth_gam <- bam( # using bam() instead of gam() for faster estimation due to large
  fantasy_points ~ positionFactor + s(ageCentered20, by = positionFactor) + years_of_experience + 
  data = player_stats_seasonal_offense_subsetDepth,
  nthreads = num_cores
)

pointsPerSeasonDepth_gamSummary <- summary(pointsPerSeasonDepth_gam)

pointsPerSeasonDepth_gamSummary
```

```
Family: gaussian
```

```
Link function: identity

Formula:
fantasy_points ~ positionFactor + s(ageCentered20, by = positionFactor) +
    years_of_experience + s(player_idFactor, ageCentered20, bs = "re")

Parametric coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)            4.789     11.706   0.409  0.68249
positionFactorQB     114.505     12.030   9.518  < 2e-16 ***
positionFactorRB      52.697     11.625   4.533 5.96e-06 ***
positionFactorTE      28.034     11.783   2.379  0.01739 *
positionFactorWR      42.614     11.279   3.778  0.00016 ***
years_of_experience    1.435      1.146   1.252  0.21063
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                                     edf  Ref.df      F  p-value
s(ageCentered20):positionFactorFB   1.578   1.966  0.799    0.402
s(ageCentered20):positionFactorQB   4.946   6.097 19.208  < 2e-16 ***
s(ageCentered20):positionFactorRB   4.202   5.179 19.582  < 2e-16 ***
s(ageCentered20):positionFactorTE   3.788   4.743  7.390 2.63e-06 ***
s(ageCentered20):positionFactorWR   4.851   5.862 23.399  < 2e-16 ***
s(player_idFactor,ageCentered20)  643.054 900.000  5.404  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.559   Deviance explained = 61.4%
fREML =  30017  Scale est. = 3196.3    n = 5369
```

```r
pointsPerSeasonDepth_gamSummary$r.sq
```

```
[1] 0.5588756
```

```r
MuMIn::r.squaredGLMM(pointsPerSeasonDepth_gam)
```

```
           R2m       R2c
[1,] 0.5340878 0.5340878
```

```r
AIC(pointsPerSeasonDepth_gam)
```

```
[1] 59192.65
```

12.3.5 Bayesian Mixed Models

12.3.5.1 Determine Response Distribution

```
player_stats_seasonal_offense_subset$fantasyPointsPosNoZeros <- player_stats_seasonal_offense_subs
player_stats_seasonal_offense_subset$fantasyPointsPos[player_stats_seasonal_offense_subset$fantasy
player_stats_seasonal_offense_subset$fantasyPointsPosNoZeros[player_stats_seasonal_offense_subset$

fantasyPointsVector <- player_stats_seasonal_offense_subset$fantasy_points %>%
    na.omit() %>%
    as.vector()

fantasyPointsVectorPosNoZeros <- fantasyPointsVectorPos <- fantasyPointsVector
fantasyPointsVectorPos[fantasyPointsVector < 0] <- 0
fantasyPointsVectorPosNoZeros[fantasyPointsVector <= 0] <- 0.01
```

```
ggplot2::ggplot(
  data = player_stats_seasonal_offense_subset,
  mapping = aes(
    x = fantasy_points)
) +
  geom_histogram(
    aes(y = after_stat(density)),
    color = "#000000",
    fill = "#0099F8"
  ) +
  geom_density(
    color = "#000000",
    fill = "#F85700",
    alpha = 0.6 # add transparency
  ) +
  geom_rug() +
  theme_classic()
```

Figure 12.27 Histogram of Fantasy Points with Overlaid Density Plot and Rug Plot.

```
fitdistrplus::descdist(fantasyPointsVector)
```

```
summary statistics
------
min:  -7.28   max:  479.46
median:  31.5
mean:  60.85402
estimated sd:  73.53479
estimated skewness:  1.788103
estimated kurtosis:  6.463928
```

Figure 12.28 Cullen and Frey Graph.

```
# all values
fit.norm <- fitdistrplus::fitdist(fantasyPointsVector, "norm")

# positive-only
fit.exp <- fitdist(fantasyPointsVectorPos, "exp")

# positive and no zeros
fit.gamma <- fitdistrplus::fitdist(fantasyPointsVectorPosNoZeros, "gamma")
fit.lognormal <- fitdistrplus::fitdist(fantasyPointsVectorPosNoZeros, "lnorm")
fit.weibull <- fitdistrplus::fitdist(fantasyPointsVectorPosNoZeros, "weibull")

# Model fit
AIC(fit.norm)
```

```
[1] 154719.7
```

```
AIC(fit.exp)
```

```
[1] 138264.5
```

```
AIC(fit.gamma) # fits best
```

```
[1] 133507.3
```

```
AIC(fit.lognormal)
```

```
[1] 139502.7
```

```
AIC(fit.weibull)
```

```
[1] 134157.1
```

```
plot(fit.norm)
```

Figure 12.29 Fit of normal distribution to fantasy points.

```
plot(fit.lognormal)
```

Figure 12.30 Fit of log normal distribution to fantasy points.

```
plot(fit.gamma)
```

Figure 12.31 Fit of gamma distribution to fantasy points.

```
plot(fit.exp)
```

Figure 12.32 Fit of exponential distribution to fantasy points.

```
plot(fit.weibull)
```

Figure 12.33 Fit of Weibull distribution to fantasy points.

```
model_normal <- glm(
  formula = fantasy_points ~ 1,
  family = gaussian(),
  data = player_stats_seasonal_offense_subset,
  maxit = 100000)

model_gamma <- glm(
  formula = fantasyPointsPosNoZeros ~ 1,
  family = Gamma(),
  data = player_stats_seasonal_offense_subset,
  maxit = 100000)

sum(resid(model_normal)^2)
```

```
[1] 73167060
```

```
sum(resid(model_gamma)^2)
```

```
[1] 32129.53
```

```
AIC(model_normal)
```

```
[1] 154719.7
```

```
AIC(model_gamma)
```

```
[1] 133996.7
```

Because the response distribution (i.e., fantasy points) is positively skewed, we will use a gamma response distribution.

12.3.5.2 Prepare Data

```
player_stats_seasonal_offense_subset$fantasy_points_posOnly <- player_stats_seasonal_offense_subse
player_stats_seasonal_offense_subset$fantasy_points_posOnly[which(player_stats_seasonal_offense_su
```

12.3.5.3 Specify Model Formula

Information about smooth terms in the `mgcv` package is provided at the following link: https://stat.ethz.ch/R-manual/R-devel/library/mgcv/html/smooth.terms.html.

Specify model formula:

```
bayesianMixedModelFormula <- brms::bf(
  fantasy_points_posOnly ~ positionFactor + s(ageCentered20, by = positionFactor) + years_of_exper
)
```

12.3.5.4 Run Model

Now, we can run the model.

> **Note 7: Bayesian mixed model**
>
> Note: the following code takes a while to run.

```
bayesianMixedModelFit <- brms::brm(
  formula = bayesianMixedModelFormula,
  data = player_stats_seasonal_offense_subset,
  family = hurdle_gamma(),
  cores = 4,
  save_pars = save_pars(latent = FALSE, all = FALSE),
  threads = threading(parallelly::availableCores()),
  backend = "cmdstanr",
  seed = 52242,
```

```
  silent = 0
)
```

12.3.5.5 Model Summary

```
summary(bayesianMixedModelFit)
```

```
 Family: hurdle_gamma
  Links: mu = log; shape = identity; hu = identity
Formula: fantasy_points_posOnly ~ positionFactor + s(ageCentered20, by = positionFactor) + years_of_ex
  Data: player_stats_seasonal_offense_subset (Number of observations: 6899)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Smoothing Spline Hyperparameters:
                                Estimate Est.Error l-95% CI u-95% CI Rhat
sds(sageCentered20positionFactorFB_1)    2.95    1.71    0.40    7.24 1.00
sds(sageCentered20positionFactorQB_1)    1.31    0.69    0.44    3.17 1.00
sds(sageCentered20positionFactorRB_1)    1.75    0.80    0.68    3.74 1.00
sds(sageCentered20positionFactorTE_1)    2.16    1.00    0.88    4.69 1.00
sds(sageCentered20positionFactorWR_1)    1.91    0.77    0.87    3.89 1.00
sds(sageCentered20player_idFactor_1)    1.96    0.20    1.59    2.36 1.01
                                      Bulk_ESS Tail_ESS
sds(sageCentered20positionFactorFB_1)     1554      931
sds(sageCentered20positionFactorQB_1)     2537     2794
sds(sageCentered20positionFactorRB_1)     1825     2235
sds(sageCentered20positionFactorTE_1)     2067     2037
sds(sageCentered20positionFactorWR_1)     2637     2857
sds(sageCentered20player_idFactor_1)       286      641

Multilevel Hyperparameters:
~player_idFactor (Number of levels: 1562)
            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.93      0.03     0.87     1.00 1.00     1218     2163

Regression Coefficients:
                               Estimate Est.Error l-95% CI u-95% CI Rhat
Intercept                          1.42      0.21     1.01     1.83 1.00
positionFactorQB                   1.97      0.21     1.56     2.38 1.00
positionFactorRB                   1.33      0.20     0.94     1.74 1.00
positionFactorTE                   0.79      0.21     0.39     1.19 1.00
positionFactorWR                   1.16      0.20     0.78     1.55 1.00
years_of_experience                0.17      0.02     0.13     0.22 1.00
```

```
sageCentered20:positionFactorFB_1   -9.62     8.96  -30.26    7.08 1.00
sageCentered20:positionFactorQB_1   -8.29     2.31  -12.80   -3.50 1.00
sageCentered20:positionFactorRB_1  -10.55     4.35  -20.50   -2.21 1.00
sageCentered20:positionFactorTE_1   -9.11     4.90  -19.56   -0.86 1.00
sageCentered20:positionFactorWR_1   -9.99     4.41  -18.88   -1.55 1.00
                                  Bulk_ESS Tail_ESS
Intercept                             1498     2016
positionFactorQB                      1503     2107
positionFactorRB                      1372     1900
positionFactorTE                      1451     2093
positionFactorWR                      1465     2165
years_of_experience                   1892     1863
sageCentered20:positionFactorFB_1     2653     2053
sageCentered20:positionFactorQB_1     2994     2499
sageCentered20:positionFactorRB_1     2569     2106
sageCentered20:positionFactorTE_1     2390     1546
sageCentered20:positionFactorWR_1     2572     2431

Further Distributional Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
shape     1.32      0.03     1.27     1.37 1.00     1749     2449
hu        0.04      0.00     0.03     0.04 1.00     8006     2740

Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
brms::prior_summary(bayesianMixedModelFit)
```

```
                prior     class                                     coef
                (flat)         b
               (flat)        b                          positionFactorQB
               (flat)        b                          positionFactorRB
               (flat)        b                          positionFactorTE
               (flat)        b                          positionFactorWR
              (flat)       b          sageCentered20:positionFactorFB_1
              (flat)       b          sageCentered20:positionFactorQB_1
              (flat)       b          sageCentered20:positionFactorRB_1
              (flat)       b          sageCentered20:positionFactorTE_1
              (flat)       b          sageCentered20:positionFactorWR_1
               (flat)        b                       years_of_experience
             beta(1, 1)        hu
 student_t(3, 3.7, 2.5) Intercept
   student_t(3, 0, 2.5)        sd
   student_t(3, 0, 2.5)        sd
```

```
  student_t(3, 0, 2.5)       sd                                Intercept
   student_t(3, 0, 2.5)       sds
  student_t(3, 0, 2.5)     sds     s(ageCentered20, by = positionFactor)
  student_t(3, 0, 2.5)     sds s(ageCentered20, player_idFactor, bs = "re")
      gamma(0.01, 0.01)     shape
           group resp dpar nlpar lb ub       source
                                             default
                                        (vectorized)
                                        (vectorized)
                                        (vectorized)
                                        (vectorized)
                                        (vectorized)
                                        (vectorized)
                                        (vectorized)
                                        (vectorized)
                                        (vectorized)
                                        (vectorized)
                                    0  1      default
                                              default
                                    0         default
 player_idFactor                    0    (vectorized)
 player_idFactor                    0    (vectorized)
                                    0         default
                                    0    (vectorized)
                                    0    (vectorized)
                                    0         default
```

12.3.5.6 Trace Plots

```
plot(bayesianMixedModelFit, ask = FALSE)
```

Figure 12.34 Trace Plots from Bayesian Mixed Model.

Figure 12.35 Trace Plots from Bayesian Mixed Model.

Figure 12.36 Trace Plots from Bayesian Mixed Model.

Figure 12.37 Trace Plots from Bayesian Mixed Model.

12.3.5.7 Posterior Predictive Check

```
pp_check(bayesianMixedModelFit) +
  ggplot2::xlim(0, 600)
```

Figure 12.38 Posterior Predictive Check from Bayesian Mixed Model.

12.3.6 Plots of Model-Implied Fantasy Points by Position and Age

```
# From Quadratic Model: All Players
pointsPerSeason_positionAge_newData$fantasyPoints_quadratic <- predict(
  object = pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperience,
  newdata = pointsPerSeason_positionAge_newData,
  re.form = NA
)

# From Quadratic Model: Players at Top of End-of-Season Depth Chart
pointsPerSeason_positionAge_newData$fantasyPoints_depthQuadratic <- predict(
  object = pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperience,
  newdata = pointsPerSeason_positionAge_newData,
  re.form = NA
```

```
)

# From GAM Model: All Players
pointsPerSeason_positionAge_newData$fantasyPoints_gam <- predict(
  object = pointsPerSeason_gam,
  newdata = pointsPerSeason_positionAge_newData,
  newdata.guaranteed = TRUE,
  exclude = "s(player_idFactor,ageCentered20)"
)

# From GAM Model: Players at Top of End-of-Season Depth Chart
pointsPerSeason_positionAge_newData$fantasyPoints_depthGAM <- predict(
  object = pointsPerSeasonDepth_gam,
  newdata = pointsPerSeason_positionAge_newData,
  newdata.guaranteed = TRUE,
  exclude = "s(player_idFactor,ageCentered20)"
)
```

Plots of model-implied fantasy points by position and age are in Figures 12.39–12.42.

12.3.6.1 Quadratic Model

```
ggplot2::ggplot(
  data = pointsPerSeason_positionAge_newData,
  mapping = aes(
    x = age,
    y = fantasyPoints_quadratic,
    color = positionFactor
  )
) +
  geom_smooth() +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position",
    subtitle = "Quadratic Model with All Players",
    color = "Position"
  ) +
  theme_classic()
```

Figure 12.39 Plot of Model-Implied Quadratic Trajectories of Fantasy Points by Age.

12.3.6.2 Quadratic Model: Top of Depth Chart

```
ggplot2::ggplot(
  data = pointsPerSeason_positionAge_newData,
  mapping = aes(
    x = age,
    y = fantasyPoints_depthQuadratic,
    color = positionFactor
  )
) +
  geom_smooth() +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position",
    subtitle = "Quadratic Model with Players Who Were Once at Top of End-of-Season Depth Chart",
    color = "Position"
  ) +
  theme_classic()
```

Figure 12.40 Plot of Model-Implied Quadratic Trajectories of Fantasy Points by Age For Players Who Were Once at the Top of the End-of-Season Depth Chart.

12.3.6.3 Generalized Additive Model

```
ggplot2::ggplot(
  data = pointsPerSeason_positionAge_newData,
  mapping = aes(
    x = age,
    y = fantasyPoints_gam,
    color = positionFactor
  )
) +
  geom_smooth() +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position",
    subtitle = "Generalized Additive Model with All Players",
    color = "Position"
  ) +
  theme_classic()
```

Figure 12.41 Plot of Implied Trajectories of Fantasy Points by Age from a Generalized Additive Model.

12.3.6.4 Generalized Additive Model: Top of Depth Chart

```
ggplot2::ggplot(
  data = pointsPerSeason_positionAge_newData,
  mapping = aes(
    x = age,
    y = fantasyPoints_depthGAM,
    color = positionFactor
  )
) +
  geom_smooth() +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age and Position",
    subtitle = "Generalized Additive Model with Players Who Were Once at Top of End-of-Season Depth
    color = "Position"
  ) +
  theme_classic()
```

Figure 12.42 Plot of Implied Trajectories of Fantasy Points by Age, from a Generalized Additive Model, For Players Who Were Once at the Top of the End-of-Season Depth Chart.

12.3.7 Plots of Individuals' Model-Implied Fantasy Points by Age

```
player_stats_seasonal_offense_subsetCC$fantasyPoints_quadratic <- predict(
  object = pointsPerSeason_positionAgeRandomLinearFixedQuadraticSlopesInteractionExperience,
  newdata = player_stats_seasonal_offense_subsetCC
)

player_stats_seasonal_offense_subsetCC$fantasyPoints_gam <- predict(
  object = pointsPerSeason_gam,
  newdata = player_stats_seasonal_offense_subsetCC
)
```

```
zeroAge <- pointsPerSeason_positionAge_newData %>%
  group_by(positionFactor) %>%
  filter(fantasyPoints_gam < 0) %>%
  slice(which.min(age))

peakAge <- pointsPerSeason_positionAge_newData %>%
  group_by(positionFactor) %>%
```

```
  slice(which.max(fantasyPoints_gam))

peakAge2 <- pointsPerSeason_positionAge_newData %>%
  filter(age > 22) %>%
  group_by(positionFactor) %>%
  slice(which.max(fantasyPoints_gam))

qbPeakAge <- round(peakAge$age[which(peakAge$positionFactor == "QB")], 0)
fbPeakAge <- round(peakAge$age[which(peakAge$positionFactor == "FB")], 0)
rbPeakAge <- round(peakAge$age[which(peakAge$positionFactor == "RB")], 0)
wrPeakAge <- round(peakAge$age[which(peakAge$positionFactor == "WR")], 0)
wrPeakAge2 <- round(peakAge2$age[which(peakAge$positionFactor == "WR")], 0)
tePeakAge <- round(peakAge$age[which(peakAge$positionFactor == "TE")], 0)

qbZeroAge <- round(zeroAge$age[which(zeroAge$positionFactor == "QB")], 0)
fbZeroAge <- round(zeroAge$age[which(zeroAge$positionFactor == "FB")], 0)
rbZeroAge <- round(zeroAge$age[which(zeroAge$positionFactor == "RB")], 0)
wrZeroAge <- round(zeroAge$age[which(zeroAge$positionFactor == "WR")], 0)
teZeroAge <- round(zeroAge$age[which(zeroAge$positionFactor == "TE")], 0)
```

12.3.7.1 Quarterbacks

A plot of Quarterbacks' model-implied fantasy points by age is in Figure 12.43. The model-implied peak of Quarterbacks' fantasy points is at age 20. The model-predicted value of zero fantasy points for Quarterbacks is at 36.

```
plot_individualFantasyPointsByAgeQB <- ggplot(
  data = player_stats_seasonal_offense_subsetCC %>% filter(position == "QB"),
  mapping = aes(
    x = age,
    y = fantasyPoints_gam,
    group = player_id)) +
  geom_smooth(
    aes(
      x = age,
      y = fantasyPoints_gam,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    se = FALSE,
    linewidth = 0.5,
    color = "black") +
  geom_smooth(
    mapping = aes(
```

```
      x = age,
      y = fantasyPoints_gam
    ),
    data = pointsPerSeason_positionAge_newData %>% filter(positionFactor == "QB"),
    inherit.aes = FALSE,
    se = TRUE,
    linewidth = 2
  ) +
  geom_point(
    aes(
      x = age,
      y = fantasyPoints_gam,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    size = 1,
    color = "transparent" # make points invisible but keep tooltips
  ) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age: Quarterbacks"
  ) +
  theme_classic()

ggplotly(
  plot_individualFantasyPointsByAgeQB,
  tooltip = c("age","fantasyPoints_gam","text","label")
)
```

Figure 12.43 Plot of Individuals' Implied Trajectories of Fantasy Points by Age, from a Generalized Additive Model, for Quarterbacks. Overlaid with the Model-Implied Trajectory for Quarterbacks in Blue.

12.3.7.2 Fullbacks

A plot of Fullbacks' model-implied fantasy points by age is in Figure 12.44. The model-implied peak of Fullbacks' fantasy points is at age 27. The model-predicted value of zero fantasy points for Fullbacks is at 31.

```
plot_individualFantasyPointsByAgeFB <- ggplot(
  data = player_stats_seasonal_offense_subsetCC %>% filter(position == "FB"),
  mapping = aes(
    x = age,
    y = fantasyPoints_gam,
    group = player_id)) +
  geom_smooth(
    aes(
      x = age,
      y = fantasyPoints_gam,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    se = FALSE,
    linewidth = 0.5,
    color = "black") +
  geom_smooth(
    mapping = aes(
      x = age,
      y = fantasyPoints_gam
    ),
    data = pointsPerSeason_positionAge_newData %>% filter(positionFactor == "FB"),
    inherit.aes = FALSE,
    se = TRUE,
    linewidth = 2
  ) +
  geom_point(
    aes(
      x = age,
      y = fantasyPoints_gam,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    size = 1,
    color = "transparent" # make points invisible but keep tooltips
  ) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
```

```
    title = "Fantasy Points (Season) by Player Age: Fullbacks"
  ) +
  theme_classic()

ggplotly(
  plot_individualFantasyPointsByAgeFB,
  tooltip = c("age","fantasyPoints_gam","text","label")
)
```

Figure 12.44 Plot of Individuals' Implied Trajectories of Fantasy Points by Age, from a Generalized Additive Model, for Fullbacks. Overlaid with the Model-Implied Trajectory for Fullbacks in Blue.

12.3.7.3 Running Backs

A plot of Running Backs' model-implied fantasy points by age is in Figure 12.45. The model-implied peak of Running Backs' fantasy points is at age 20. The model-predicted value of zero fantasy points for Running Backs is at 30.

```
plot_individualFantasyPointsByAgeRB <- ggplot(
  data = player_stats_seasonal_offense_subsetCC %>% filter(position == "RB"),
  mapping = aes(
    x = age,
    y = fantasyPoints_gam,
    group = player_id)) +
  geom_smooth(
    aes(
      x = age,
      y = fantasyPoints_gam,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    se = FALSE,
    linewidth = 0.5,
    color = "black") +
  geom_smooth(
    mapping = aes(
      x = age,
      y = fantasyPoints_gam
    ),
    data = pointsPerSeason_positionAge_newData %>% filter(positionFactor == "RB"),
    inherit.aes = FALSE,
    se = TRUE,
    linewidth = 2
  ) +
  geom_point(
    aes(
      x = age,
      y = fantasyPoints_gam,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    size = 1,
    color = "transparent" # make points invisible but keep tooltips
  ) +
  labs(
    x = "Player Age (years)",
```

```
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age: Running Backs"
  ) +
  theme_classic()

ggplotly(
  plot_individualFantasyPointsByAgeRB,
  tooltip = c("age","fantasyPoints_gam","text","label")
)
```

Figure 12.45 Plot of Individuals' Implied Trajectories of Fantasy Points by Age, from a Generalized Additive Model, for Running Backs. Overlaid with the Model-Implied Trajectory for Running Backs in Blue.

12.3.7.4 Wide Receivers

A plot of Wide Receivers' model-implied fantasy points by age is in Figure 12.46. The model-implied peaks of Wide Receivers' fantasy points are at ages 20 and 26. The model-predicted value of zero fantasy points for Wide Receivers is at 30.

```
plot_individualFantasyPointsByAgeWR <- ggplot(
  data = player_stats_seasonal_offense_subsetCC %>% filter(position == "WR"),
  mapping = aes(
    x = age,
    y = fantasyPoints_gam,
    group = player_id)) +
  geom_smooth(
    aes(
      x = age,
      y = fantasyPoints_gam,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    se = FALSE,
    linewidth = 0.5,
    color = "black") +
  geom_smooth(
    mapping = aes(
      x = age,
      y = fantasyPoints_gam
    ),
    data = pointsPerSeason_positionAge_newData %>% filter(positionFactor == "WR"),
    inherit.aes = FALSE,
    se = TRUE,
    linewidth = 2
  ) +
  geom_point(
    aes(
      x = age,
      y = fantasyPoints_gam,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    size = 1,
    color = "transparent" # make points invisible but keep tooltips
  ) +
  labs(
    x = "Player Age (years)",
```

```
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age: Wide Receivers"
  ) +
  theme_classic()

ggplotly(
  plot_individualFantasyPointsByAgeWR,
  tooltip = c("age","fantasyPoints_gam","text","label")
)
```

Figure 12.46 Plot of Individuals' Implied Trajectories of Fantasy Points by Age, from a Generalized Additive Model, for Wide Receivers. Overlaid with the Model-Implied Trajectory for Wide Receivers in Blue.

12.3.7.5 Tight Ends

A plot of Tight Ends' model-implied fantasy points by age is in Figure 12.47. The model-implied peak of Tight Ends' fantasy points is at age 20. The model-predicted value of zero fantasy points for Tight Ends is at 32.

```
plot_individualFantasyPointsByAgeTE <- ggplot(
  data = player_stats_seasonal_offense_subsetCC %>% filter(position == "TE"),
  mapping = aes(
    x = age,
    y = fantasyPoints_gam,
    group = player_id)) +
  geom_smooth(
    aes(
      x = age,
      y = fantasyPoints_gam,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    se = FALSE,
    linewidth = 0.5,
    color = "black") +
  geom_smooth(
    mapping = aes(
      x = age,
      y = fantasyPoints_gam
    ),
    data = pointsPerSeason_positionAge_newData %>% filter(positionFactor == "TE"),
    inherit.aes = FALSE,
    se = TRUE,
    linewidth = 2
  ) +
  geom_point(
    aes(
      x = age,
      y = fantasyPoints_gam,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    size = 1,
    color = "transparent" # make points invisible but keep tooltips
  ) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
```

```
    title = "Fantasy Points (Season) by Player Age: Tight Ends"
  ) +
  theme_classic()

ggplotly(
  plot_individualFantasyPointsByAgeTE,
  tooltip = c("age","fantasyPoints_gam","text","label")
)
```

Figure 12.47 Plot of Individuals' Implied Trajectories of Fantasy Points by Age, from a Generalized Additive Model, for Tight Ends. Overlaid with the Model-Implied Trajectory for Wide Tight Ends in Blue.

12.3.8 Summary of Findings

We applied mixed models with random intercepts and random slopes to allow our model estimates to account for the fact that different players have different starting points (intercepts) and different changes over time (slopes) in fantasy points. A quadratic, inverted-U-shaped form as a function of age fit better than a linear form as a function of age in predicting players' fantasy points. A generalized additive model that allowed further nonlinearity fit even better than the quadratic model.

Based on the bivariate scatterplots between age and fantasy points, we might conclude that players tend to stay stable or even increase in fantasy points with age. However, this conclusion would be wrong. When we account for the longitudinal data (i.e., multiple observations over time for the same player) using mixed models, we observe that fantasy points tend to decrease with age, with the timing and rate of decline differing for each position. In other words, the association between age and fantasy points differs at the person level versus the group level. This is an example of Simpson's paradox.

The discrepancy between the positive or null association between age and players' fantasy points at the group level, and the negative association at the person level may be due, in part, to the selective attrition of players with age. The players who play the longest will tend to be the highest performing players, whereas the poorest performing players will retire or get dropped from the team at younger ages. Thus, the selective attrition of weaker players may make it seem that there is no association between age and performance (or even a positive one!), when in fact, players' performance tends to decrease with age after age 26 or so (with the timing differing from position to position), until the player eventually retires or is dropped from the team. Selective attrition is common in longitudinal studies (such as this one) and intervention studies. For instance, attrition may be more likely for individuals from lower socioeconomic status backgrounds because they may face more challenges in continuing in longitudinal studies such as fewer financial resources, greater life stressors, etc. In addition, people who experience side effects or lack of improvement may be more likely to drop out of intervention studies. Examining only those who completed treatment (an example of selection bias) would make the intervention look more effective than it actually was because the people who stay in the study are those who experience the greatest improvement. Thus, it is important to use approaches such as mixed models or other approaches that account for the multiple observations from the same person, that use all available information, and that do not exclude people who do not complete all portions of the study.

12.4 Conclusion

Mixed models allow accounting for multiple levels or units of analysis and to include both fixed and random effects. Inclusion of random effects allows the association between the predictor variables (the intercept and age) and the outcome variable (fantasy points) to differ for each individual in the grouping level (in this case, each player). This allows for more accurately predicting phenomena. Based on the bivariate scatterplots between age and fantasy points, we might conclude that players tend to stay stable or even increase in fantasy points with age. However, this conclusion would be wrong. When we account for the longitudinal data using mixed models, we observe that players' fantasy points tend to decrease with age, with the timing and rate of decline differing for each position. In other words, the association between age and fantasy points differs at the person level versus the group level, which is an example of Simpson's paradox. In sum, mixed models are valuable for examining associations between variables when there are multiple levels of data (i.e., multiple observations within the same unit, known as clustering or nesting). It is important not to confuse the association at one level (e.g., group level) with the association at another level (e.g., person level), which is an example of the ecological fallacy.

13

Causal Inference

13.1 Getting Started

13.1.1 Load Packages

```
library("dagitty")
library("ggdag")
```

13.2 Correlation Does Not Imply Causation

As described in Section 8.3.2.1, there are several reasons why two variables, `X` and `Y`, might be correlated:

- `X` causes `Y`
- `Y` causes `X`
- `X` and `Y` are bidirectional: `X` causes `Y` and `Y` causes `X`
- a third variable (i.e., confound), `Z`, influences both `X` and `Y`
- the association between `X` and `Y` is spurious

13.3 Criteria for Causality

How do we know whether two processes are causally related? There are three criteria for establishing causality (Shadish et al., 2002):

1. The cause (e.g., the independent or predictor variable) temporally precedes the effect (i.e., the dependent or outcome variable).

2. The cause is related to (i.e., associated with) the effect.
3. There are no other alternative explanations for the effect apart from the cause.

The first criterion for establishing causality involves temporal precedence. In order for a cause to influence an effect, the cause must occur before the effect. For instance, if sports drink consumption influences player performance, the sports drink consumption (that is presumed to influence performance) must occur prior to the performance improvement. Establishing the first criterion eliminates the possibility that the association between the purported cause and effect reflects reverse causation. Reverse causation occurs when the purported effect is actually the cause of the purported cause, rather than the other way around. For instance, if sports drink consumption occurs only once, and it occurs only before and not after performance, then we have ruled out the possibility of reverse causation (i.e., that better performance causes players to consume sports drink).

The second criterion involves association. The purported cause must be associated with the purported effect. Nevertheless, as the maxim goes, "correlation does not imply causation." Just because two variables are correlated does not necessarily mean that they are causally related. However, correlation is useful because causality requires that the two processes be correlated. That is, correlation is a necessary but insufficient condition for causality. For instance, if sports drink consumption influences player performance, sports drink consumption must be associated with performance improvement.

The third criterion involves ruling out alternative reasons why the purported cause and effect may be related. As noted in Section 13.2, there are four reasons why `X` may be correlated with `Y`. If we meet the first criterion of causality, we have removed the possibility that `Y` causes `X` (i.e., reverse causality). To meet the third criterion of causality, we need to remove the possibility that the association reflects a third variable (confound) that influences both the cause and effect, and we need to remove the possibility that the association is spurious—the possibility that the association between the purported cause and effect is due to random chance.

There are multiple approaches to meeting the third criterion of causality, such as by use of experiments, longitudinal designs, control variables, within-subject designs, and genetically informed designs, as described in Section 13.4.

In general, to meet the third criterion of causality, one must consider the counterfactual. A *counterfactual* is what would have happened in the hypothetical scenario that the cause did not occur [i.e., what would have happened in the absence of the cause; Shadish et al. (2002)]. When engaging in causal inference, it is important to consider what would have happened if the hypothetical cause had actually not occurred. For instance, consider that we conduct an experiment to randomly assign some players to consume a sports

drink before a game and other players to drink only water. In this case, our treatment/intervention group is the group of players that consumed a sports drink. The control group is the group players that drank only water. Now, consider that the players in the treatment group outperform the players in the the control group in their football game. In such a study, we observe what *did happen* when players received a treatment. The counterfactual is knowledge of what *would have happened* to those same players if they simultaneously had not received treatment (Shadish et al., 2002). The true causal effect, then, is the difference between what did happen and what would have happened. However, we cannot observe a counterfactual. That is, we do not know for sure what would have happened to the players who received treatment if those same players had actually not received treatment. We have a control group, but the control group does not have the same players as the intervention group, and it is impossible for a person to simultaneously receive and not receive treatment.

So, our goal in working toward causal inference as scientists is to create reasonable approximations to this impossible counterfactual (Shadish et al., 2002). For instance, if using a between-subject design, we want the two groups to be equivalent in every possible way except whether or not they receive the treatment, so we might stratify each group to be equivalent in terms of age, weight, position, experience, skill, etc. Or, we might test the same people using an A-B-A-B within-subject design. In an A-B-A-B within-subject design, players receive no treatment at baseline (timepoint 1: game 1), receive the treatment at timepoint 2 (game 2), receive no treatment at timepoint 3 (game 3), and receive the treatment at timepoint 4 (game 4). Neither of these approximations is a true counterfactual. In the between-subject design, the players differ between the two groups, so we cannot know how the individuals who received the treatment would have performed if they had actually not received the treatment. In the A-B-A-B within-subject design, the players are the same, but they timepoints that they receive or do not receive the treatment differ, and there can be carryover effects from one condition to the next. For instance, consuming sports drinks before game 2 might also help them be better hydrated in general, including, for subsequent games. Thus, we cannot know how a player would have performed in game 1 with treatment or in game 2 without treatment, etc. Nevertheless, it is important to be aware of the counterfactual and to engage in counterfactual reasoning to consider what would have happened if the supposed cause had not occurred. Considering the counterfactual is important for designing closer approximations to the counterfactual in studies for stronger research designs and stronger causal inference.

13.4 Approaches for Causal Inference

13.4.1 Experimental Designs

As described in Section 8.3.1, experimental designs are designs in which participants are randomly assigned to one or more levels of the independent variable to observe its effects on the dependent variable. Experimental designs provide the strongest tests of causality because they can rule out reverse causation and third variables. For instance, by manipulating sports drink consumption before the player performs, they can eliminate the possibility that reverse causation explains the effect of the independent variable on the dependent variable. Second, through randomly assigning players to consume or not consume sports drink, this holds everything else constant (so long as the groups are evenly distributed according to other factors, such as their age, weight, etc.) and thus removes the possibility that third variable confounds explain the effect of the independent variable on the dependent variable.

13.4.2 Quasi-Experimental Designs

Although experimental designs provide the strongest tests of causality, manytimes they are impossible, unethical, or impractical to conduct. For instance, it would likely not be practical to randomly assign National Football League (NFL) players to either consume or not consume sports drink before their games. Players have their pregame rituals and routines and many would likely not agree to participate in such a study. Thus, we often rely on quasi-experimental designs such as natural experiments and observational/correlational designs.

We cannot directly test or establish causality from a non-experimental research design. Nevertheless, we can leverage various design features that, in combination with other studies using different research methods, collectively strengthen our ability to make causal inferences. For instance, there are are no experiments in humans showing that smoking causes cancer—randomly assigning people to smoke or not smoke would not be ethical. The causal inference that smoking causes cancer was derived from a combination of experimental studies in rodents and observational studies in humans.

13.4.2.1 Longitudinal Designs

Research designs can be compared in terms of their internal validity—the extent to which we can be confident about causal inferences. A cross-sectional association is depicted in Figure 13.1:

Figure 13.1 Cross-Sectional Association. T1 = Timepoint 1. From Petersen (2024b) and Petersen (2024c).

For instance, we might observe that sports drink consumptions is concurrently associated with better player performance. Among observational/correlational research designs, cross-sectional designs tend to have the weakest internal validity. For the reasons described in Section 13.2, if we observe a cross-sectional association between `X` (e.g., sports drink consumption) and `Y` (e.g., player performance), we have little confidence that `X` causes `Y`. As a result, longitudinal designs can be valuable for more closely approximating causality if an experimental designs is not possible. Consider a lagged association that might be observed in a longitudinal design, as in Figure 13.2, which is a slightly better approach than relying on cross-sectional associations:

For instance, we might observe that sports drink performance *before* the game is associated with better player performance *during* the game. A lagged association has somewhat better internal validity than a cross-sectional association because we have greater evidence of temporal precedence—that the influence of the predictor *precedes* the outcome because the predictor was assessed before the outcome and it shows a predictive association. However, part of the association between the predictor with later levels of the outcome could be due to prior levels of the outcome that are stable across time. That is, it could be that better player performance leads players to consume more sports drink and

Figure 13.2 Lagged Association. T1 = Timepoint 1. T2 = Timepoint 2. From Petersen (2024b) and Petersen (2024c).

that player performance is relatively stable across time. In such a case, it may be observed that sports drink consumption predicts later player performance even though player performance influences sports drink consumption, rather than the other way around Thus, consider an even stronger alternative—a lagged association that controls for prior levels of the outcome, as in Figure 13.3:

For instance, we might observe that sports drink performance *before* the game is associated with better player performance *during* the game, while controlling for prior player performance. A lagged association controlling for prior levels of the outcome has better internal validity than a lagged association that does not control for prior levels of the outcome. A lagged association that controls for prior levels further reduces the likelihood that the association owes to the reverse direction of effect, because earlier levels of the outcome are controlled. However, consider an even stronger alternative—lagged associations

Figure 13.3 Lagged Association, Controlling for Prior Levels of the Outcome. T1 = Timepoint 1. T2 = Timepoint 2. From Petersen (2024b) and Petersen (2024c).

that control for prior levels of the outcome and that simultaneously test each direction of effect, as depicted in Figure 13.4:

Lagged associations that control for prior levels of the outcome and that simultaneously test each direction of effect provide the strongest internal validity among observational/correlational designs. Such a design can help better clarify which among the variables is the chicken and the egg—which variable is more likely to be the cause and which is more likely to be the effect. If there are bidirectional effects, such a design can also help clarify the magnitude of each direction of effect. For instance, we can simultaneously evaluate the extent to which sports drink predicts later player performance (while controlling for prior performance) and the reverse—player performance predicting later sports drink consumption (while contorlling for prior sports drink consumption).

Figure 13.4 Lagged Association, Controlling for Prior Levels of the Outcome, Simultaneously Testing Both Directions Of Effect. T1 = Timepoint 1. T2 = Timepoint 2. From Petersen (2024b) and Petersen (2024c).

13.4.2.2 Within-Subject Analyses

Another design feature of longitudinal designs that can lead to greater internal validity is the use of within-subject analyses. Between-subject analyses, might examine, for instanc, whether players who consume more sports drink perform better on average compared to players who consume less sports drink. However, there are other between-person differences that could explain any observed between-subject associations between sports drink consumption and players performance. Another approach could be to apply within-subject analyses. For instance, you could examining whether, within the same individual, if a player consumes a sports drink, do they perform better compared to games in which they did not consume a sports drink. When we control for prior levels of the outcome in the prediction, we are evaluating whether the predictor is

associated with witin-person *change* in the outcome. Predicting within-person change provides stronger evidence consistent with causality because it uses the individual as their own control and controls for many time-invariant confounds (i.e., confounds that do not change across time). However, predicting within-person change does not, by itself, control for time-varying confounds. So, it can also be useful to control for time-varying confounds, such as by use of control variables.

13.4.2.3 Control Variables

One of the plausible alternatives to the inference that `X` causes `Y` is that there are third variable confounds that influence both `X` and `Y`, thus explaining why `X` and `Y` are associated, as depicted in Figures 8.3 and 13.10. Thus, another approach that can help increase internal validity is to include plausible confounds as control variables. For instance, if a third variable such as education level might be a confound that influences both sports drink consumption and player performance, you could include education level as a covariate in the model. Inclusion of a covariate attempts to control for the variable by examining the association between the predictor variable and the outcome variable while holding the covariate variables constant. For instance, such a model would examine whether, when accounting for education level, there is an association between sports drink consumption and player performance.

Failure to control for important third variables can lead to erroneous conclusions, as evidenced by the association depicted in Figure 13.5. In the example, if we did not control for gender, we would infer that there is a positive association between dosage and recovery probability. However, when we examine each men and women separately, we learn that the association between dosage and recovery probability is actually negative within each gender group. Thus, in this case, failure to control for gender would lead to false inferences about the association between dosage and recovery probability.

However, it can be problematic to control for variables indiscriminantly (Spector & Brannick, 2010; Wysocki et al., 2022). The use of causal diagrams can inform which variables are important to be included as control variables, and—just as important—which variables not to include as control variables, as described in Section 13.5.

13.4.2.4 Genetically Informed Designs

Another approach to control for variables is to use genetically informed designs. Genetically informed designs allow controlling for potential genetic effects in order to more closely approximate the contributions of various environmental effects. Genetically informed designs exploit differing degrees of genetic relatedness among participants to capture the extent to

Figure 13.5 Example Where Failing to Control for a Variable (In This Case, Gender) Would Lead to False Inferences. In this example, the association between dosage and recovery probability is positive at the population level, but the association is negative among men and women separately. (Figure reprinted from Kievit et al. (2013), Figure 1, p. 2. Kievit, R., Frankenhuis, W., Waldorp, L., & Borsboom, D. (2013). Simpson's paradox in psychological science: A practical guide. *Frontiers in Psychology*, *4*(513). https://doi.org/10.3389/fpsyg.2013.00513)

which genetic factors may contribute to an outcome. The average percent of DNA shared between people of varying relationships is provided in Table 13.1 (https://isogg.org/wiki/Autosomal_DNA_statistics; archived at https://perma.cc/MK3D-DST8):

Table 13.1 Average Percent of Autosomal DNA Shared by Pairs of Relatives by Relationship Type.

Relationship	Average Percent of Autosomal DNA Shared by Pairs of Relatives
Monozygotic ("identical") twins	100%
Dizygotic ("fraternal") twins	50%
Parent/child	50%
Full siblings	50%
Grandparent/grandchild	25%
Aunt-or-uncle/niece-or-nephew	25%
Half-siblings	25%
First cousin	12.5%
Great-grandparent/great-grandchild	12.5%

For instance, researchers may compare monozygotic twins versus dizygotic twins in some outcome—a so-called "twin study". It is assumed that the trait/outcome is attributable to genetic factors to the extent that the monozygotic twins (who share 100% of their DNA) are more similar in the trait or outcome compared to the dizygotic twins (who share on average 50% of their DNA). Alternatively, researchers could compare full siblings versus half-siblings, or they could compare full siblings versus first cousins.

Genetically informed designs are not as relevant for fantasy football analytics, but they are useful to present as one of various design features that researchers can draw upon to strengthen their ability to make causal inferences.

13.5 Causal Diagrams

13.5.1 Overview

A key tool when describing a research question or hypothesis is to create a conceptual depiction of the hypothesized causal processes. A causal diagram depicts the hypothesized causal processes that link two or more variables. A common form of causal diagrams is the directed acyclic graph (DAG). DAGs provide a helpful tool to communicate about causal questions and help identify how to avoid bias (i.e., overestimation) in associations between variables due to confounding (i.e., common causes) (Digitale et al., 2022). For instance, from a DAG, it is possible to determine what variables it is important to control for in order to get unbiased estimates of the association between two variables of interest. To create DAGs, you can use the `R` package `dagitty` (Textor et al., 2017) or the associated browser-based extension, DAGitty: https://dagitty.net (archived at https://perma.cc/U9BY-VZE2). Examples of various causal diagrams that could explain why `X` is associated with `Y` are in Figures 13.6, 13.8 and 13.10.

```
XCausesY <- dagitty::dagitty("dag{
  X -> Y
}")

plot(dagitty::graphLayout(XCausesY))

dagitty::impliedConditionalIndependencies(XCausesY)
```

Figure 13.6 Causal Diagram (Directed Acyclic Graph) Depicting `X` Causing `Y`.

Here is an alternative way of specifying the same diagram (more similar to `lavaan` syntax):

```
XCausesY_alt <- ggdag::dagify(
  Y ~ X
)

#plot(XCausesY_alt) # this creates the same plot as above
ggdag::ggdag(XCausesY_alt) + theme_dag_blank()
```

Figure 13.7 Causal Diagram (Directed Acyclic Graph) Depicting `X` Causing `Y`.

```
YCausesX <- dagitty::dagitty("dag{
  Y -> X
}")

plot(dagitty::graphLayout(YCausesX))

dagitty::impliedConditionalIndependencies(YCausesX)
```

Figure 13.8 Causal Diagram (Directed Acyclic Graph) Depicting Reverse Causation: `Y` Causing `X`.

Here is an alternative way of specifying the same diagram (more similar to `lavaan` syntax):

```
YCausesX_alt <- ggdag::dagify(
  X ~ Y
)

#plot(YCausesX_alt) # this creates the same plot as above
ggdag::ggdag(YCausesX_alt) + theme_dag_blank()
```

Figure 13.9 Causal Diagram (Directed Acyclic Graph) Depicting Reverse Causation: `Y` Causing `X`.

```
ZCausesXandY <- dagitty::dagitty("dag{
  Z -> Y
  Z -> X
  X <-> Y
}")

plot(dagitty::graphLayout(ZCausesXandY))
```

Figure 13.10 Causal Diagram (Directed Acyclic Graph) Depicting a Third Variable Confound, `Z`, Causing `X` and `Y`, Thus Explaining Why `X` and `Y` are associated.

Here is an alternative way of specifying the same diagram (more similar to `lavaan` syntax):

```
ZCausesXandY_alt <- ggdag::dagify(
  X ~ Z,
  Y ~ Z,
  X ~~ Y
)

#plot(ZCausesXandY_alt) # this creates the same plot as above
ggdag::ggdag(ZCausesXandY_alt) + theme_dag_blank()
```

Figure 13.11 Causal Diagram (Directed Acyclic Graph) Depicting a Third Variable Confound, `Z`, Causing `X` and `Y`, Thus Explaining Why `X` and `Y` are associated.

Consider another example in Figure 13.12:

```
mediationDag <- dagitty::dagitty("dag{
  X -> M1
  X -> M2
  M1 -> Y
  M2 -> Y
  M1 <-> M2
}")

plot(dagitty::graphLayout(mediationDag))
```

Figure 13.12 Causal Diagram (Directed Acyclic Graph).

```
dagitty::impliedConditionalIndependencies(mediationDag)
```

```
X _||_ Y | M1, M2
```

```
dagitty::adjustmentSets(
  mediationDag,
  exposure = "M1",
  outcome = "Y",
  effect = "total")
```

```
{ M2 }
```

In this example, `X` influences `Y` via `M1` and `M2` (i.e., multiple mediators), and `M1` is also associated with `M2`. The `dagitty::impliedConditionalIndependencies()` function identifies variables in the causal diagram that are conditionally independent (i.e., uncorrelated) after controlling for other variables in the model. For this causal diagram, `X` is conditionally independent with `Y` because `X` is independent with `Y` when controlling for `M1` and `M2`.

The `dagitty::adjustmentSets()` function identifies variables that would be necessary to control for (i.e., to include as covariates) in order to identify an unbiased estimate of the association (whether the total effect, i.e., `effect = "total"`; or the direct effect, i.e., `effect = "direct"`) between two variables

(`exposure` and `outcome`). In this case, to identify the unbiased association between `M1` and `Y`, it is important to control for `M2`.

Here is an alternative way of specifying the same diagram (more similar to `lavaan` syntax):

```
mediationDag_alt <- ggdag::dagify(
  M1 ~ X,
  M2 ~ X,
  Y ~ M1,
  Y ~ M2,
  M1 ~~ M2
)

#plot(mediationDag_alt) # this creates the same plot as above
ggdag::ggdag(mediationDag_alt) + theme_dag_blank()
```

Figure 13.13 Causal Diagram (Directed Acyclic Graph).

13.5.2 Confounding

Confounding occurs when two variables—that are both caused by another variable(s)—have a spurious or noncausal association (D'Onofrio et al., 2020). That is, two variables share a common cause, and the common cause leads the variables to be associated even though they are not causally related. The

common cause—i.e., the variable that influences the two variables—is known as a confound (or confounder). An example of confounding is depicted in Figure 13.14:

```
confounding <- ggdag::confounder_triangle(
  x = "Player Endurance",
  y = "Field Goals Made",
  z = "Stadium Altitude")

confounding %>%
  ggdag(
    text = FALSE,
    use_labels = "label") +
  theme_dag_blank()
```

Figure 13.14 Causal Diagram (Directed Acyclic Graph) Example of Confounding.

```
dagitty::impliedConditionalIndependencies(confounding)
```

```
x _||_ y | z
```

The output indicates that player endurance (`X`) and field goals made (`Y`) are conditionally independent when accounting for stadium altitude (`Z`). *Conditional independence* refers to two variables being unassociated when controlling for other variables.

```
dagitty::adjustmentSets(
  confounding,
  exposure = "x",
  outcome = "y",
  effect = "total")
```

```
{ z }
```

The output indicates that, to obtain an unbiased estimate of the causal association between two variables, it is necessary to control for any confounding (Lederer et al., 2019). That is, to obtain an unbiased estimate of the causal association between player endurance (`X`) and field goals made (`Y`), it is necessary to control for stadium altitude (`Z`).

13.5.3 Mediation

Mediation can be divided into two types: full and partial. In full mediation, the mediator(s) fully account for the effect of the predictor variable on the outcome variable. In partial mediation, the mediator(s) partially but do not fully account for the effect of the predictor variable on the outcome variable.

13.5.3.1 Full Mediation

An example of full mediation is depicted in Figure 13.15:

```
full_mediation <- ggdag::mediation_triangle(
  x = "Coaching Quality",
  y = "Fantasy Points",
  m = "Player Preparation")

full_mediation %>%
  ggdag(
    text = FALSE,
    use_labels = "label") +
  theme_dag_blank()
```

Figure 13.15 Causal Diagram (Directed Acyclic Graph) Example of Full Mediation.

```
dagitty::impliedConditionalIndependencies(full_mediation)
```

```
x _||_ y | m
```

In full mediation, `X` and `Y` are conditionally independent when accounting for the mediator (`Z`). In this case, coaching quality (`X`) and fantasy points (`Y`) are conditionally independent when accounting for player preparation (`M`). In other words, in this example, player preparation is the mechanism that fully (i.e., 100%) accounts for the effect of coaching quality on players' fantasy points.

```
dagitty::adjustmentSets(
  full_mediation,
  exposure = "x",
  outcome = "y",
  effect = "direct")
```

```
{ m }
```

The output indicates that, to obtain an unbiased estimate of the *direct* causal association between coaching quality (`X`) and fantasy points (`Y`) (i.e., the effect that is *not* mediated through intermediate processes), it is necessary to control for player preparation (`M`).

```
dagitty::adjustmentSets(
  full_mediation,
  exposure = "x",
  outcome = "y",
  effect = "total")
```

```
{}
```

However, to obtain an unbiased estimate of the *total* causal association between coaching quality (`X`) and fantasy points (`Y`) (i.e., including the portion of the effect that is mediated through intermediate processes), it is important *not* to control for player preparation (`M`). When the goal is to understand the (total) causal effect of coaching quality (`X`) and fantasy points (`Y`), controlling for the mediator (player preparation; `M`) would be inappropriate because doing so would remove the causal effect, thus artificially reducing the estimate of the association between coaching quality (`X`) and fantasy points (`Y`) (Lederer et al., 2019).

13.5.3.2 Partial Mediation

An example of partial mediation is depicted in Figure 13.16:

```
partial_mediation <- ggdag::mediation_triangle(
  x = "Coaching Quality",
  y = "Fantasy Points",
  m = "Player Preparation",
  x_y_associated = TRUE)

partial_mediation %>%
  ggdag(
    text = FALSE,
    use_labels = "label") +
  theme_dag_blank()
```

Figure 13.16 Causal Diagram (Directed Acyclic Graph) Example of Partial Mediation.

```
dagitty::impliedConditionalIndependencies(partial_mediation)
```

In partial mediation, `X` and `Y` are *not* conditionally independent when accounting for the mediator (`Z`). In this case, coaching quality (`X`) and fantasy points (`Y`) are still associated when accounting for player preparation (`M`). In other words, in this example, player preparation is a mechanism that partially but does not fully account for the effect of coaching quality on players' fantasy points. That is, there are likely other mechanisms, in addition to player preparation, that collectively account for the effect of coaching quality on fantasy points. For instance, coaching quality could also influence player fantasy points through better play-calling.

```
dagitty::adjustmentSets(
  partial_mediation,
  exposure = "x",
  outcome = "y",
  effect = "direct")
```

```
{ m }
```

As with full mediation, the output indicates that, to obtain an unbiased estimate of the *direct* causal association between coaching quality (`X`) and fantasy

points (Y) (i.e., the effect that is *not* mediated through intermediate processes), it is necessary to control for player preparation (M).

```
dagitty::adjustmentSets(
  partial_mediation,
  exposure = "x",
  outcome = "y",
  effect = "total")
```

```
{}
```

However, as with full mediation, to obtain an unbiased estimate of the *total* causal association between coaching quality (X) and fantasy points (Y) (i.e., including the portion of the effect that is mediated through intermediate processes), it is important *not* to control for player preparation (M). When the goal is to understand the (total) causal effect of coaching quality (X) and fantasy points (Y), controlling for a mediator (player preparation; M) would be inappropriate because doing so would remove the causal effect, thus artificially reducing the estimate of the association between coaching quality (X) and fantasy points (Y) (Lederer et al., 2019).

13.5.4 Ancestors and Descendants

In a causal model, an *ancestor* is a variable that influences another variable, either directly or indirectly via another variable (Rohrer, 2018). A *descendant* is a variable that is influenced by another variable (Rohrer, 2018). In general, one should not control for descendants of the outcome variable, because doing so could eliminate the apparent effect of a legitimate cause on the outcome variable (Digitale et al., 2022). Moreover, as described above, when trying to understand the total causal effect of a predictor variable on an outcome variable, one should not control for descendants of the predictor variable that are also antecedents of the outcome variable (i.e., mediators of the effect of the predictor variable on the outcome variable) (Digitale et al., 2022).

Consider the example in Figure 13.17:

```
descendentDag <- dagitty::dagitty("dag{
  X -> M
  M -> Y
  X -> Y
  Y -> Z
}")

#plot(dagitty::graphLayout(descendentDag))
ggdag::ggdag(descendentDag) + theme_dag_blank()
```

Figure 13.17 Causal Diagram (Directed Acyclic Graph) Example of Ancestor (Mediation) and Descendant of Y.

```
dagitty::impliedConditionalIndependencies(descendentDag)
```

```
M _||_ Z | Y
X _||_ Z | Y
```

In this example, `X` and `M` are conditionally independent with `Z` when accounting for the mediator (`Y`).

```
dagitty::adjustmentSets(
  descendentDag,
  exposure = "X",
  outcome = "Y",
  effect = "direct")
```

```
{ M }
```

```
dagitty::adjustmentSets(
  descendentDag,
  exposure = "X",
  outcome = "Y",
  effect = "total")
```

```
{}
```

As indicated above, one should not control for the descendant of the outcome variable; thus, one should not control for `Z` when examining the association between `X` or `M` and `Y`.

13.5.5 Collider Bias

Collision occurs when two variables influence a third variable, the collider (D'Onofrio et al., 2020). That is, a collider is a variable that is caused by two other variables (i.e., a common effect). An example collision is depicted in Figures 13.18 and 13.19:

```
colliderBias1 <- ggdag::collider_triangle(
  x = "Diet",
  y = "Coaching Strategy",
  m = "Injury Status")

colliderBias1 %>%
  ggdag(
    text = FALSE,
    use_labels = "label") +
  theme_dag_blank()
```

Figure 13.18 Causal Diagram (Directed Acyclic Graph) Example of a Collision with a Collider (Injury Status).

```
dagitty::impliedConditionalIndependencies(colliderBias1)
```

```
x _||_ y
```

In this example collision, diet (`X`) and coaching strategy (`Y`) are independent.

```
dagitty::adjustmentSets(
  colliderBias1,
  exposure = "x",
  outcome = "y",
  effect = "total")
```

```
 {}
```

As the output indicates, we should not control for the collider when examining the association between the two causes of the collider. That is, we should not control for injury status (`M`) when examining the association between diet (`X`) and coaching strategy. Controlling for the collider leads to confounding and can artificially induce an association between the two causes of the collider despite no causal association between them (Lederer et al., 2019). Obtaining a distorted or artificial association between two variables due to inappropriately controlling for a collider is known as *collider bias*.

Consider another example:

```
colliderBias2 <- ggdag::collider_triangle(
  x = "Coaching Quality",
  y = "Player Preparation",
  m = "Fantasy Points",
  x_y_associated = TRUE)

colliderBias2 %>%
  ggdag(
    text = FALSE,
    use_labels = "label") +
  theme_dag_blank()
```

Figure 13.19 Causal Diagram (Directed Acyclic Graph) Example of Collider Bias.

```
dagitty::impliedConditionalIndependencies(colliderBias2)
```

In this example of collider bias, there are no conditional independencies.

```
dagitty::adjustmentSets(
  colliderBias2,
  exposure = "x",
  outcome = "y",
  effect = "total")
```

```
{}
```

Again, it would be important not to control for the collider, fantasy points (`M`), when examining the association between coaching quality (`X`) and player preparation (`Y`). In this case, controlling for the collider would remove some of the causal effect of coaching quality on player preparation and could lead to an artificially smaller estimate of the causal effect between coaching quality and player preparation.

13.5.5.1 M-Bias

Collider bias may also occur when neither variable of interest is a direct cause of the collider (Lederer et al., 2019). M-bias is a form of collider bias that occurs when two variables that are not causally related, `A` and `B`, both influence a collider, `M`, and each (`A` and `B`) also influences a separate variable—e.g., `A` influences `X` and `B` influences `Y`. M-bias is so-named from the M-shape of the DAG. An example of M-bias is depicted in Figure 13.20:

```
mBias <- ggdag::m_bias(
  x = "Number of Media Articles About the Team",
  y = "Fantasy Points",
  a = "Team Record",
  b = "Coaching Quality",
  m = "Fan Attendance at Game")

mBias %>%
  ggdag(
    text = FALSE,
    use_labels = "label") +
  theme_dag_blank()
```

Figure 13.20 Causal Diagram (Directed Acyclic Graph) Example of M-Bias.

In this example, fan attendance is the collider that is influenced separately by the team record and the coaching quality. This is a fictitious example for purposes of demonstration; in reality, coaching quality influences the team's record.

```
dagitty::impliedConditionalIndependencies(mBias)
```

```
a _||_ b
a _||_ y
b _||_ x
m _||_ x | a
m _||_ y | b
x _||_ y
```

As the output indicates, there are several conditional independencies.

```
dagitty::adjustmentSets(
  mBias,
  exposure = "x",
  outcome = "y",
  effect = "total")
```

```
{}
```

It is important not to control for the collider (fan attendance). If you control for the collider, you can induce an artificial association between team record and coaching quality. Moreover, because doing so induces an artificial association between team record and coaching quality, it can also induce an artificial association between the effects of team record and coaching quality: number of media articles about the team and fantasy points, respectively. That is, controlling for the collider can lead to an artificial association between `X` and `Y` that does not reflect a causal process.

13.5.5.2 Butterfly Bias

Butterfly bias occurs when both confounding and M-bias are present. Butterfly bias (aka bow-tie bias) is so-named from the butterfly shape of the DAG. In butterfly bias, the following criteria are met:

- Two variables (`A` and `B`) influence a collider (`M`).
- The collider influences two variables, `X` and `Y`.
- `A` also influences `X`.
- `B` also influences `Y`.
- `A` and `B` are not causally related.
- `X` and `Y` are not causally related.

Or, more succinctly:

- `A` influences `M` and `X`.
- `B` influences `M` and `Y`.
- `M` influences `X` and `Y`.

In butterfly bias, the collider (`M`) is also a confound. That is, a variable is both influenced by two variables and influences two variables. An example of butterfly bias is depicted in Figure 13.21:

```
butterflyBias <- ggdag::butterfly_bias(
  x = "Off-field Behavior",
  y = "Fantasy Points",
  a = "Diet",
  b = "Coaching Quality",
  m = "Mental Health")

butterflyBias %>%
  ggdag(
    text = FALSE,
    use_labels = "label") +
  theme_dag_blank()
```

Figure 13.21 Causal Diagram (Directed Acyclic Graph) Example of Butterfly Bias.

In this case, players' mental health is a collider of their diet and the quality of the coaching they receive. In addition, players' mental health is a confound of their off-field behavior and fantasy points.

```
dagitty::impliedConditionalIndependencies(butterflyBias)
```

```
a _||_ b
a _||_ y | b, m
b _||_ x | a, m
x _||_ y | b, m
x _||_ y | a, m
```

As the output indicates, there are several conditional independencies.

```
dagitty::adjustmentSets(
  butterflyBias,
  exposure = "x",
  outcome = "y",
  effect = "total")
```

```
{ b, m }
{ a, m }
```

When dealing with a collider that is also a confound, controlling for either set, `B` and `M` or `A` and `M`, will provide an unbiased estimate of the association between `X` and `Y`. In this case, controlling for either a) coaching quality and mental health or b) diet and mental health—but not both sets—will yield an unbiased estimate of the association between off-field behavior and fantasy points.

13.5.6 Selection Bias

Selection bias occurs when the selection of participants or their inclusion in analyses depends on the variables being studied. For instance, if you are conducting a study on the extent to which sports drink consumption influences fantasy points, there would be selection bias if players are less likely to participate in the study if they score fewer fantasy points.

Now, consider a study in which you conduct a randomized controlled trial (RCT; i.e., an experiment) to evaluate the effect of a new medication on player performance. You randomly assign some players to take the medication and other players to take a placebo. Assume the new medication has side effects and leads many of the players who take it to drop out of the study. This is an example of attrition bias (i.e., systematic attrition). If you were to exclude these individuals from your analysis, it may make it appear that the medication led to better performance, because the players who experienced the side effect (and worse performance) dropped out of the study. Hence, conducting an analysis that excludes these players from the analysis would involve selection bias.

13.6 Conclusion

There are three criteria for establishing causality: 1) the cause precedes the effect. 2) The cause is related to the effect. 3) There are no other alternative explanations for the effect apart from the cause. In general, it is important to be aware of the counterfactual and to consider what would have happened if the supposed cause had not occurred. Various experimental and quasi-experimental designs and approaches can be leveraged to more closely approximate causal inferences. Longitudinal designs, within-subject analyses, inclusion of control variables, and genetically informed designs are all quasi-experimental designs that afford the researcher greater control over some possible third variable confounds. Causal diagrams can be a useful tool for identifying the proper variables to control for (and those not to control for). When confounding exists, it is important to control for the confound(s). It is im-

portant not to control for mediators when interested in the total effect of the predictor variable on the outcome variable. In addition, it is important not to control for descendants of the outcome variable. When there is a collision, it is important not to control for the collider (unless the collider is also a confound).

14

Heuristics and Cognitive Biases in Prediction

14.1 Getting Started

14.1.1 Load Packages

```
library("tidyverse")
```

14.2 Overview

When considering judgment and prediction, it is important to consider psychological concepts, including heuristics and cognitive biases. In the modern world of big data, research and society need people who know how to make sense of the information around us. Given humans' cognitive biases, it is valuable to leverage more objective approaches than relying on our "gut" and intutition. Statistical approaches can be a more objective way to identify systematic patterns.

Statistical analysis—and science more generally—is a process to the pursuit of knowledge. An *epistemology* is an approach to knowledge. Science is perhaps the best approach (epistemology) that society has to approximate truth. Unlike other approaches to knowledge, science relies on empirical evidence and does not give undue weight to anecdotal evidence, intuition, tradition, or authority.

Per Petersen (2024c), here are the characteristics of science that distinguish it from pseudoscience:

1. Risky hypotheses are posed that are falsifiable. The hypotheses can be shown to be wrong.
2. Findings can be replicated independently by different research groups and different methods. Evidence converges across studies and methods.
3. Potential alternative explanations for findings are specified and examined empirically (with data).
4. Steps are taken to guard against the undue influence of personal beliefs and biases.
5. The strength of claims reflects the strength of evidence. Findings and the ability to make judgments or predictions are not overstated. For instance, it is important to present the degree of uncertainty from assessments with error bars or confidence intervals.
6. Scientifically supported measurement strategies are used based on their psychometrics, including reliability and validity.

Nevertheless, statistical analysis is not purely objective and is not a panacea. Science is a human enterprise—it is performed by humans each of whom has their own biases. For instance, cognitive biases can influence how people interpret statistics. As a result, the findings from any given study may be incorrect. Thus, it would be imprudent to make decisions based solely on the results of one study. That is why we wait for findings to be independently replicated by different groups of researchers using different methods.

If a research team publishes flashy new and exciting findings, other researchers have an incentive to disprove the prior findings. Thus, we have more confidence if findings stand up to scrutiny from independent groups of researchers. We also draw upon meta-analyses—studies of many studies, to summarize the results of many studies and not just the findings from any single study that may not replicate. In this way, we can identify which findings are robust and most likely true versus the findings that fail to replicate. Thus, despite its flaws like any other human enterprise, science is a self-correcting process in which the long arc bends toward truth.

In our everyday lives, humans are presented with overwhelming amounts of information. Because human minds cannot parse every piece of information equally, we tend to take mental shortcuts, called *heuristics*. These mental shortcuts can be helpful. They reduce our mental load and can help us make quick judgments to stay alive or to make complex decisions in the face of uncertainty. However, these mental shortcuts can also lead us astray and to make systematic errors in our judgments and predictions. *Cognitive biases* are

systematic errors in thinking. *Fallacies* are forms of flawed reasoning.

14.3 Examples of Heuristics

As described above, heuristics are mental shortcuts that people use to handle the overwhelming amount of information to process. Important heuristics used in judgment and prediction in the face of uncertainty include (Critcher & Rosenzweig, 2014; Kahneman, 2011; Tversky & Kahneman, 1974):

- availability heuristic
- representativeness heuristic
- anchoring and adjustment heuristic
- affect heuristic
- performance heuristic
- "What You See Is All There Is" (WYSIATI) heuristic

Use of heuristics has been observed among fantasy sports managers (Smith et al., 2006).

14.3.1 Availability Heuristic

The *availability heuristic* refers to the tendency for a person's judgments or predictions about the frequency or probability of something to be made based on how readily instances can be brought to mind. For instance, when making fantasy predictions about a player, more recent big performance games may more easily come to mind compared to lower-scoring games and games that occurred longer ago. Thus, a manager may be more inclined to pick players to start who had more recent, stronger performances rather than players who have higher long-term averages.

14.3.2 Representativeness Heuristic

The *representativess heuristic* refers to the tendency for a person's judgments or predictions about individuals to be made based on how similar the individual is to (i.e., how closely the individual resembles) the person's existing mental prototypes. For instance, when coming out of college, Tight End Kyle Pitts drew comparisons[1] to the "LeBron James" of Tight Ends (archived at

[1] https://247sports.com/article/kyle-pitts-lebron-james-2021-nfl-draft-florida-gators-football-163882176

https://perma.cc/JQB5-XPVL). The idea that his athletic profile leads him to be similar to the prototype of LeBron James may have led him to be too highly drafted by fantasy managers in his first seasons.

The representativeness heuristic has been observed in gambling markets for predicting team wins in the National Football League (NFL) (Woodland & Woodland, 2015) and in decision making in fantasy soccer (Kotrba, 2020).

14.3.3 Anchoring and Adjustment Heuristic

The *anchoring and adjustment heuristic* refers to the tendency for a person's judgments or predictions to be made with a reference point—an anchor—as a starting point from which they adjust their estimates upward or downward. The anchor is often inaccurate and given too much weight in the person's calculation, and too little adjustment is made to the anchor. For instance, a manager is trying to predict how many fantasy points a top Running Back may score. The player scored 300 fantasy points last season, but the team added a stronger backup Running Back and changed the Offensive Coordinator to be a more pass-heavy offense. The manager may use 300 fantasy points as an anchor (based on the player's performance last season), and may adjust downward 15 points to account for the offseason changes. However, it is possible that this downward adjustment is insufficient to account not only for the offseasons changes but also for potential regression effects. Regression effects are discussed further in Section 14.5.2.

14.3.4 Affect Heuristic

The *affect heuristic* refers to the tendency for a person to make judgments or predictions based on their emotions and feelings—such as whether they like or dislike something—with little deliberation or reasoning (Kahneman, 2011). For instance, if a manager's favorite team is the Dallas Cowboys, they may hold a distaste for players on teams that are rivals of the Cowboys, such as the Philadelphia Eagles. The affect heuristic may lead them to avoid drafting Eagles players even though some Eagles players may perform well.

14.3.5 Performance Heuristic

The *performance heuristic* refers to the tendency for people to predict their improvement in a task based on their previous performance (Critcher & Rosenzweig, 2014). Although past performance tends to be correlated with future performance, people tend to show regression to the mean, as described in Section 14.5.2. Thus, people who show strong performance in their first few games may in many cases show worse performance in subsequent games. Moreover,

initial performance does not necessarily indicate the extent of one's future *improvement*. For instance, if a manager has a successful first season, the performance heuristic may lead them to overestimate the extent to which they will improve in their skill or performance in the future.

14.3.6 What You See Is All There Is (WYSIATI) Heuristic

The "What You See Is All There Is" (WYSIATI) heuristic refers to the tendency for a person to make a decision or judgment based solely on the information that is immediately available to them (i.e., "right in front of them"), without considering the possibility that there may be additional important information that they do not have (Enke, 2020). The WYSIATI heuristic involves jumping to conclusions on the basis of limited information (Kahneman, 2011). For instance, a manager might start a player based on the fact that the player scored lots of points in their most recent game. However, the manager may not consider other important factors that they did not know about, such as the fact that the player filled in for an injured player, and will get fewer touches in the upcoming game now that the original starter is back to full health.

14.4 Examples of Cognitive Biases

As described above, cognitive biases are systematic errors in thinking. They lead us to be less accurate in our judgments and predictions. Inaccuracy may not always be bad; several cognitive biases may serve to help people feel better about themselves (R. M. Miller, 2013), including confirmation bias, overconfidence bias, optimism bias, and self-serving bias. In fantasy football, however, cognitive biases and inaccurate judgments can lead to worse performance. Cognitive biases are often due to the use of heuristics.

Miller (2013) describes cognitive biases in fantasy sports. Examples of cognitive biases relevant to fantasy football that result from one or more heuristics include:

- overconfidence bias
- optimism bias
- confirmation bias
- in-group bias
- hindsight bias
- outcome bias

- self-serving bias
- omission bias
- loss aversion bias
- primacy effect bias
- recency effect bias
- framing effect bias
- endowment effect bias
- bandwagon effect bias
- Dunning–Kruger effect bias

14.4.1 Overconfidence Bias

In general, people tend to be overconfident in their judgments and predictions. *Overconfidence bias* is the tendency for a person to have greater confidence in their abilities (including judgments and predictions) than is objectively warranted. There are three general ways that overconfidence has been identified (Moore & Healy, 2008):

1. *overestimation* of one's actual performance
2. *overplacement* of one's performance relative to others
3. *overprecision* in one's beliefs/jugments/predictions

Overestimation involves believing that one will perform better than one actually performs. Overestimation can be identified with a calibration plot of the predicted performance versus actual performance, where the person's predicted performance is systematically higher (in at least some cases) than their actual performance. Overestimation corresponds to the "overprediction" form of miscalibration.

Overplacement involves believing that one is better than others or will perform better than others, even when they do not. For instance, it is a common finding that more than half of people believe they are "above average" (i.e., above the median), even though that is statistically impossible. This calls to mind the fictitious Lake Wobegon in the radio show *A Prariri Home Companion*, "where all the women are strong, all the men are good-looking, and all the children are above average."

Overprecision involves expressing excessive certainty regarding the accuracy of one's beliefs/judgments/predictions. For instance, if when a given meteorologist says it will rain 80% of the time, it actually rains 30% of the time, the meteorologist's predictions are overprecise. Likewise, if the weather forecast says it will rain 10% of the time and it actually rains 30% of the time, the predictions are also overprecise because the forecaster is expressing stronger confidence than is warranted that it will not rain. Overprecision can be identified with a calibration plot of the predicted probabilities versus the actual

probabilities. Overprecision corresponds to the "overextremity" form of miscalibration.

Overestimation and overprecision are studied in various ways. Typically, people are asked about (a) whether an event will occur or (b) the likelihood that the event will occur, across many events. For the former (approach "a"), people may be asked to make a dichotomous judgment or prediction, by responding to the question: e.g., "Will it rain tomorrow? [YES/NO]". They will then rate their confidence (as a percentage) in their answer (0–100%). They would make each of these two ratings for each of many events. Then, we can evaluate, for a given respondent, the degree to which the probabilistic estimate of an event reflects the true underlying probability of the event. For instance, for a given respondent (and for respondents in general), for the events when the respondent says they are 80% confident an event (e.g., rain) will occur, does the event actually occur around 80% of the time? For the latter (approach "b"), people may indicate the likelihood that the event will occur, by responding to the question: "How likely is it that it will rain tomorrow? (0–100%)". Then, we can evaluate, for instance, for the events when the respondent says that an event (e.g., rain) is 80% likely to occur, does the event actually occur around 80% of the time?

The confidence that people tend to have in their predictions depends less and the quantity and quality of the evidence for their predictions and more on the quality of the narrative they can tell about the (little) information they have immediate access to (Kahneman, 2011). For instance, people tend not to consider other relevant, important factors that they do not have immediate access to, consistent with the WYSIATI heuristic.

A fantasy manager may be even more likely to exhibit overconfidence if they previously performed well or won their league, for which luck and random chance plays an important role. Indeed, it is estimated that nearly half (~45%) of the variability in fantasy football performance is estimated to be luck [and around 55% due to skill; Getty et al. (2018)]. A manager who won their league in the prior season may believe they will perform better than they actually will (overestimation), will perform better than average (overplacement), and may hold excessive confidence regarding the accuracy of their predictions about which players will perform well or poorly (overprecision). These various types of overconfidence may lead them to draft high-risk players based on gut feeling, neglecting statistical analysis and expert consensus.

Players' performance in fantasy football, and human behavior more generally, is complex and multiply determined (i.e., is influenced by many factors). Despite the bluster of so-called experts who pretend to know more than they can know, no one can consistently and accurately predict how all players will perform. Remain humble in your predictions; do not be more confident than is warranted. If you approach the task of prediction with humility, you may be more able to be flexible and more willing to consider other players who you

can draft for good value.

14.4.2 Optimism Bias

Optimism bias is the tendency for people to overestimate one's likelihood of experiencing positive events and underestimate one's likelihood of experiencing negative events. For instance a manager might overestimate the likelihood that their team will win the championship and may underestimate the likelihood that they may lose a player to injury.

14.4.3 Confirmation Bias

Confirmation bias is the tendency for people to search for, interpret, and remember information that confirms one's beliefs, as opposed to information that might disconfirm one's beliefs. The search for confirming evidence is called "positive test strategy"; it runs counter to advice by philosophers of science who advise testing hypotheses by trying to refute them [i.e., by searching for *disconfirming* evidence; Kahneman (2011)]. The result of confirmation bias is that people are unlikely to change their minds about something that they have a pre-existing belief about, because they tend to look only for information that supports their pre-existing beliefs. For instance, if you believe that a particular player is a strong breakout candidate to be a sleeper, you may be more likely to pay attention to evidence that supports that the player will breakout and may be less likely to pay attention to evidence that indicates the player may struggle. That is, people tend to look for confirmation that the players they want to draft (or start) are preferred by the experts (R. M. Miller, 2013). Confirmation bias may lead to "going with your gut" (i.e., making a decision based on your intuition).

As a budding empiricist, you should actively seek out information that challenges or disconfirms your beliefs—particularly from trustworthy, reputable sources—and work to incorporate the information into your beliefs. Do your best to go into observation, data analysis, and data interpretation with an open mind.

14.4.4 In-Group Bias

In-group bias is the tendency for people to prefer others who are members of their same (in-)group. For instance, a manager whose favorite team is the Dallas Cowboys might try to draft mostly Cowboys players, regardless of their historical performance.

14.4.5 Hindsight Bias

"Hindsight is 20/20." – Idiom

Hindsight bias is the tendency to perceive that past events were more predictable than they were. People tend to remember the succeess of their predictions and forget the failures of their predictions. For instance, if a third-string Quarterback has a breakout game, a fantasy manager may claim that they "knew it all along" that the player was going to breakout, despite not having picked up the player. That same manager may forget the many other predictions they had that did not come true.

14.4.6 Outcome Bias

Outcome bias is the tendency to evaluate the quality of a decision based on the eventual outcome of that decision, rather than based on the quality of the decision based on the information that was known at the time of the decision. For instance, consider that a manager starts a lower-ranked player over a higher-ranked player because the lower-ranked player has a more favorable matchup. Now, consider that the lower-ranked player scores fewer points than the higher-ranked player because the lower-ranked player got injured. If the manager concludes that their decision-making process for who to start was flawed and thus not pay attention to matchups in subsequent decisions, this would reflect outcome bias. Likewise, if the manager makes a poor-quality decision but gets lucky, and they attribute their success to a sound decision-making process, this would also reflect outcome bias.

Focus on the decision-making process when evaluating the quality of decisions. Yes, also evaluate the outcomes of decisions, but do not give yourself too much credit for lucky decisions or too much blame for unlucky decisions (like injuries). That said, one thing you can count on is that players get injured. So, it is to your benefit to draft and compose your team to have some bench depth so you can handle injuries when they inevitably occur.

14.4.7 Self-Serving Bias

Self-serving bias is the tendency to perceive oneself in an overly positive manner. Doing so may function to bolster people's self-esteem. One way that self-

serving bias can commonly manifest is the tendency people to attribute successes to internal factors—such as one's character, ability, effort, or actions—but to attribute failures to external factors beyond one's control—such as others, unfairness, bad luck, etc. That is, self-serving bias involves claiming more reponsibility for one's success than for one's failure (R. M. Miller, 2013). For instance, a manager who wins the league may attribute their success to their fantasy football skill and smart decisions (rather than luck). However, if they get last place, they may attribute their failure not to themselves but instead to factors outside of their control, such as bad luck, player injuries, and poor performance by players due to circumstances out of their control (e.g., bad weather conditions, coaching decisions).

Although the self-serving bias may help us feel better about ourselves, it can lead us to overestimate our abilities when things go well and to fail to learn from mistakes when things go poorly. Remember that nearly half of the variability in fantasy football performance is estimated to be luck, and the other half likely due to skill (Getty et al., 2018). Thus, successes in fantasy football likely involve a good amount of luck; likewise, failures in fantasy football can also reflect mistakes and poor decision making that arise from lower skill.

14.4.8 Omission Bias

Omission bias is the tendency to prefer inaction over action. For instance, consider a manager who is on the fence to drop an underperforming player and to pick up someone else to start. If the manager does not drop the player and the player scores few points, the manager is likely to be less critical of their decision than if they had picked up another player who then scored the same (low) number of points.

Consider another example adapted from Miller (2013): two players scored the same number of points in a given week. A Wide Receiver scored 1.6 points from 16 receiving yards on 1 target. A Running Back scored 1.6 points from 56 receiving yards and two dropped fumbles. The manager may harbor more blame for the player who performed a harmful action (losing two fumbles) than the player who received only one target (harmful inaction).

Pay attention to players' usage and opportunities, not just how many points a player scored and whether they had more fumbles or interceptions than other players. The more times a player touches the ball, the more points they are likely to score.

14.4.9 Loss Aversion Bias

Loss aversion bias is the tendency to avoid losses rather than acquiring equivalent gains. That is, loss aversion (similar to punishment sensitivity, as opposed to reward sensitivity) is the tendency to weigh losses (or punishment) more heavily than gains (or rewards). Loss aversion is different from risk aversion. Risk aversion is the tendency to prefer outcomes with low uncertainty (compared to outcomes with greater uncertainty). Loss aversion is exemplified when teams play conservatively so as "not to lose" instead of "to win." In fantasy football, loss aversion may lead managers to start or hold onto underperforming high drafts for too long instead of a starting a more promising player out of fear of losing potential value from their initial investment.

14.4.10 Primacy Effect Bias

Primacy effect bias is the tendency to recall the first information that one encounters better than information presented later on. For instance, primacy effect bias might lead a manager to draft a player based on the first information they encountered about the player.

14.4.11 Recency Effect Bias

Recency effect bias is the tendency to weigh recent events more than earlier ones. For instance, a manager might observe that a Running Back on the waiver wire performed well in the last two games. Recency effect bias may lead the manager to pick up the player, overvaluing their recent performance. For instance, the manager may not have adequately weighed the player's overall season performance and the fact that the starting Running Back is returning to the lineup from injury, and that is why the player received more carries in the past two games (i.e., in place of the injured starter).

Recency effect bias seems a bit at odds with primacy effect bias; however, they can both coexist. People tend to remember the earliest information they encounter in addition to the most recent information, and they tend to forget the information encountered in between.

14.4.12 Framing Effect Bias

Framing effect bias is the tendency to make decisions based on how information is framed (e.g., based on positive or negative connotations), rather than just the information itself. For instance, consider a weekly column written by two different fantasy football expert commentators. Expert A may write that a Running Back has scored at least 15 fantasy points in 70% of games this

season (i.e., positive framing). Expert B may write that the same Running Back has scored fewer than 15 fantasy points in 30% of games this season (i.e., negative framing). Both statements convey the same statistical information; however, a manager might be more inclined to start the player when reading the statement from Expert A than from Expert B because of how the information was framed (i.e., presented). People prefer to make decisions to avoid loss, consistent with loss aversion bias, so the negative framing might lead a person to be less likely to start the player.

Note how information is framed, knowing that the framing itself could lead you astray. Thus, try to focus on the information itself rather than the framing when making decisions.

14.4.13 Endowment Effect Bias

Endowment effect bias is the tendency to overvalue merely because one owns it. That is, a manager tends to value the same player more when they play for their team rather than an opponent's team. When trading, the endowment effect bias might lead a manager to demand more to trade away a player than to acquire the same player (R. M. Miller, 2013). For instance, a manager might overvalue a player they drafted in the first round, refusing to trade them even if they could get a better-performing player in return.

A player's trade value is not fixed. It makes sense to adjust a player's trade value based on many factors such as your team's needs, the needs of your trade partner's team, the player's remaining strength of schedule, etc. However, you want to avoid adjusting a player's value based merely on the fact that he is on your team. Evaluate a variety of trade analyzers to more objectively evaluate players' worth.

14.4.14 Bandwagon Effect Bias

The *bandwagon effect bias* is the tendency to do or believe things because other people are. It involves social conformity. For instance, consider if a rookie Wide Receiver has a breakout game and he is picked up in many fantasy leagues. A given manager might pick up the player because the player is frequently being picked up in many fantasy leagues, without evaluating whether the player's success is sustainable.

14.4.15 Dunning–Kruger Effect Bias

> "The more you know, the more you know you don't know." – Anonymous

The *Dunning–Kruger effect bias* is the tendency for people with low ability/competency in a task to overestimate their ability. The Dunning–Kruger effect is depicted in Figures 14.1 and 14.2. For instance, consider a new fantasy manager who experiences some initial wins (often called "beginner's luck"). They may attribute their successes to their skill rather than to luck. Their overconfidence may lead them to believe they can win the league without much preparation.

Figure 14.1 Dunning–Krueger Effect: Confidence as a Function of Competence. Adapted from https://commons.wikimedia.org/wiki/File:Effet_Dunning-Kruger.svg.

14.5 Examples of Fallacies

As described above, fallacies are mistaken beliefs and flawed reasoning. Fallacies are often due to the use of heuristics and to cognitive biases. Examples of fallacies include:

- base rate fallacy (aka base rate neglect)

Figure 14.2 Dunning–Krueger Effect: Perceived Performance as a Function of Actual Performance. Adapted from https://commons.wikimedia.org/wiki/File:Dunning-kruger_effect_-_percentile.svg.

- regression fallacy
- hot hand fallacy
- sunk cost fallacy
- gambler's fallacy
- anecdotal fallacy
- conditional probability fallacy
- ecological fallacy

14.5.1 Base Rate Fallacy

The *base rate fallacy* (aka base rate neglect) is the tendency to ignore information about the general probability of an event in favor of specific information about the event. The *base rate* is a marginal probability, which is the general probability of an event irrespective of other things. For example, the base rate of work-related injury is the general probability of experiencing a work-related injury, irrespective of other factors (e.g., the type of job, the person's age, the person's sex). Among the working population in the U.S., the life-

time prevalence of work-related injuries (i.e., the percent of people who will experience a work-related injury at some point in their lives), is ~35% (https://www.cdc.gov/mmwr/volumes/69/wr/mm6913a1.htm; archived at https://perma.cc/A2L6-WPEH). Thus, the base rate of work-related injuries in the U.S. is ~35%. The probability of work-related injuries is higher for some occupations (e.g., construction) and for some groups (e.g., men, 55–64-year-olds, Black or Multiracial, who are self-employed and have less than high school education) than others. Nevertheless, if we ignore all of the interacting factors, the general probability of work-related injuries is 35%. If we made a prediction that someone would be highly likely ($> 90\%$) to experience a work-related injury because they are male and self-employed, this would be ignoring the relatively lower base rate of work-related injury. Indeed, even men (36.7%) and self-employed individuals (41.2%) have less than a 50% chance of experiencing a work-related injury.

As applied to fantasy football, consider that you read about a potential sleeper Wide Receiver who had a stellar performance in a preseason game. If you select this player early on in the draft based on this information, this would be ignoring the general probability that most players who have a strong performance in the preseason do not perform as well in the regular season (i.e., base rate neglect). Performance in the preseason is not strongly predictive of performance in the regular season (https://fivethirtyeight.com/features/the-nfl-preseason-is-not-predictive-but-it-can-often-seem-that-way/; archived at https://perma.cc/FSG2-6AXE).

More information on base rates and how to counteract the base rate fallacy in described in Chapter 16.

14.5.2 Regression Fallacy

The regression fallacy is the failure to account for the fact that things tend to naturally fluctuate around their mean and that, after an extreme fluctuation, subsequent scores tend to regress (or reverse) to the mean. An example of the regression fallacy is the so-called Sports Illustrated or Madden cover jinx curse. The Sports Illustrated or Madden cover jinx curse is the urban legend that players who appear on the cover of Sports Illustrated (the magazine) or Madden (the video game) will perform poorly. But, such a phenomenon can be more simply explained by regression to the mean (https://www.psychologytoday.com/us/blog/what-the-luck/201610/the-sports-illustrated-cover-jinx; archived at https://perma.cc/CZM9-TVFN). When a player has a superb season, they likely benefited from some degree to good luck, and it is unlikely that they will repeat such a stellar season the following year. Instead, they are likely—at least based on random fluctuation—to regress to their long-term mean.

Applied to fantasy football, consider that a Quarterback had a 5-touchdown

game in Week 1. You are in need of a strong Quarterback, so you drop a solid player to pick him up. However, it is possible that the Quarterback benefited from playing against a week defense in the first game of the season. Future matchups may prove more difficult, and the player is unlikely to sustain such a solid performance consistently throughout the season (i.e., they are likely to regress toward their mean).

14.5.3 Hot Hand Fallacy

The "hot hand" is the idea that a player who experiences a successful outcome will have greater chance of success in subsequent attempts. For instance, in basketball, it is widely claimed by coaches, players, and commentators that players who have the hot hand (i.e., who are "on fire") are more likely to make shots because they made previous shots. Evidence on the hot hand is mixed. Considerable evidence historically has suggested that there is no such thing as a "hot hand" (Avugos et al., 2013; Bar-Eli et al., 2006; Gilovich et al., 1985). Some recent research, however, has suggested that there may be a small hot hand effect in some contexts (Bocskocsky et al., 2014; J. B. Miller & Sanjurjo, 2014). However, if any such effect exists, the hot hand may be limited to a small subset of players and the effect size of any hot hand effect appears to be small (Pelechrinis & Winston, 2022).

In football, when trying how to distribute the ball among multiple Running Backs, it is not uncommon to hear that a coach wants to give the ball to the Running Back with the "hot hand." In fantasy football, consider that a player just had a multiple touchdown game. Due to the hot hand fallacy, a manager might continue to start the player because they believe the player is "on fire" and is likely to continue to score at an unsustainable rate.

It is important consider whether such a string of strong performances are outliers and if the player may, in future games, regress to the mean. When considering whether strong performances are outliers and may regress to the mean, it is valuable to consider whether the player's health, skill, or situation has appreciably changed (compared to the player's earlier, weaker performances). Is the player finally fully healthy? Has the player appreciably improved in some skill that will benefit them in future games? Has the player's long-term situation improved, such as moving up the depth chart, or receiving more carries/targets that is not tied to a specific opponent or game script? Or, alternatively, do the improvements appear to be driven by transient, game-specific factors, such as a the health of a teammate, the opponents they played, or the game script that ensued? If long-term outlook of the player has appreciably changed due to changes in the fundamentals of a player's value, such as their health, skill, or situation, it is less likely that such performance improvements will regress over the long run.

14.5.4 Sunk Cost Fallacy

A *sunk cost* is a cost (e.g., in money, time, or effort) that has already been incurred and cannot be recovered. For instance, if a person orders an expensive meal at a restaurant, the order is a sunk cost. The *sunk cost fallacy* is the tendency to continue an endeavor when there is a sunk cost. For instance, when ordering the expensive meal at the restaurant, a person may over-eat so that they feel that they eat their money's worth of food.

Applied to fantasy football, consider a situation in which you invest a lot of salary cap or a high draft pick to draft a promising player, but they repeatedly underperform. If you continue to start the player to justify your large investment, instead of benching him in favor of a higher-performing player, you are committing the sunk cost fallacy. It is important to own up to our mistakes; when you make a mistake, the sooner you realize it, own it—for instance, by either benching or dropping the weak-performing player—and move on, the sooner you will be able to replace him with a better-performing player. At the same time, you probably do not want to give too much weight to a player's performance in any given week, due to random chance that may lead a player to greatly under- or overperform; and whose future peformance is likely to regress (whether positively or negatively) to their mean.

14.5.5 Gambler's Fallacy

The gambler's fallacy occurs due to an erroneous belief in the law of small numbers. The law of large numbers is a mathematical theorem that the average of a sufficiently large number of independent observations converges to the true value. For instance, if you flip a fair coin 1 million times, it is likely to land heads-up ~50% of the time. The law of *small* numbers (aka hasty generalization), by contrast, is an erroneous belief that small samples are representative of the populations from which they were drawn. For instance, if you flip a coin 10 times, belief in the law of small numbers would lead one to believe that the coin will flip heads-up exactly 5 times out of 10. However, in reality, the chance is less than 1 in 4 (24.6%) that exactly 5 of 10 coin flips turn up heads, as calculated below and as depicted in Figure 14.3:

```
dbinom(
  x = 5,      # number of coins that flip heads-up
  size = 10, # how many times you flip a coin
  prob = 0.5 # probability of a coin flipping heads-up (i.e., fair coin = 50%)
)
```

```
[1] 0.2460938
```

The `dbinom()` function in R provides the density of a binomial distribution. A binomial distribution is the probability of a particular number of successes (e.g., coins flipping heads-up) given a certain number of independent trials.

```
set.seed(52242)

numHeads <- rbinom(
  n = 100000,
  size = 10,
  prob = .5
)

simulationOfFlipping10Coins <- data.frame(
  numHeads = numHeads
)

simulationOfFlipping10Coins <- simulationOfFlipping10Coins %>%
  mutate(
    highlight = ifelse(numHeads == 5, "yes", "no")
  )

ggplot2::ggplot(
  data = simulationOfFlipping10Coins,
  mapping = aes(
    x = numHeads,
    fill = highlight)
) +
  geom_histogram(
    color = "#000000",
    bins = 11
  ) +
  scale_x_continuous(
    breaks = 0:10
  ) +
  scale_fill_manual(
    values = c(
      "yes" = "tomato",
      "no" = "gray")
  ) +
  labs(
    x = "Number of Coins Flipped Heads (out of 10 Coin Flips)",
    y = "Frequency",
    title = "Histogram of Number of Coins that Flip Heads-Up\nin a Simulation of 10 Coin Flips\n(w
  ) +
  theme_classic() +
```

```
  theme(legend.position = "none")
```

Figure 14.3 Histogram of Number of Coins that Flip Heads-Up in a Simulation of 10 Coin Flips (with 100,000 Replications).

Although 5 is the modal count of coins that flip heads-up out of 10 flips (i.e., it was more common than any other number), it is less common than the aggregate probability of flipping any number of heads besides 5. The probability of getting any other number of coin flips turning up heads (other than 5) is:

```
dbinom(
  x = c(0:4, 6:10),
  size = 10,
  prob = 0.5
) %>% sum()
```

```
[1] 0.7539063
```

The *gambler's fallacy* is the erroneous belief that future probabilities are influenced by past events, even when the events are independent. For example, a gambler may pay close attention to a particular slot machine. If the slot machine has not paid out in a while, the gambler may believe that the slot machine is about to pay out soon, and may start putting coins in the slots.

Applied to fantasy football, consider that a Quarterback has had several lousy games in a row. The gambler's fallacy might lead a manager to start the player under the belief that the player "is due" for a big game, expecting a strong performance from the player merely because they player has not had a good game in a while.

14.5.6 Anecdotal Fallacy

The *anecdotal fallacy* is the error of drawing inferences based on personal experience or singular examples rather than based on stronger forms of evidence. Although people tend to find stories and anecdotes compelling, they are among the weakest forms of evidence, as described in Section 8.4.3. Consider that a fantasy football commentator writes an article about how they won their league by drafting a Quarterback and Wide Receiver by drafting with their first two picks, respectively. The anecdotal fallacy might lead a manager who reads the article to believe that they are more likely to win their league if they follow a similar strategy, despite stronger evidence otherwise.

14.5.7 Conditional Probability Fallacy

We describe the conditional probability fallacy in Section 16.3.2 after introducing conditional probability in Section 16.3.1.3.

14.5.8 Ecological Fallacy

The *ecological fallacy* is the error of drawing inferences about an individual from group-level data. For instance, a manager is interested in how age is associated with fantasy performance. They examine the correlation between age and fantasy points and find that age is positively correlated with fantasy points among Quarterbacks. The ecological fallacy would then take this finding to infer that an individual Quarterback is likely to increase in their performance with age. We describe the ecological fallacy more in Section 12.2.1.

14.6 Conclusion

In conclusion, there are many heuristics, cognitive bias, and fallacies that people engage in when making judgments and predictions. It is prudent to be aware of these common biases and to work to counteract them. For instance,

look for information that challenges or disconfirms your beliefs, and work to incorporate this information into your beliefs. Do your best to pursue observation, data analysis, and data interpretation with an open mind. You never know what important information you might discover if you go in with an open mind. Pay attention to fundamentals of a player's value, such as their health, skill, or situation when considering whether a player's performance may regress to the mean. If the strong performances appear to be driven by transient, game-specific factors, such as a the health of a teammate, the opponents they played, or the game script that ensued, future performances may be more likely to regress to the mean. In general, people tend to be overconfident in their predictions. There is considerable luck in fantasy football. Approach the task of prediction with humility; no one is consistently able to accurately predict how well players will perform.

15

Judgment Versus Actuarial Approaches to Prediction

15.1 Getting Started

15.1.1 Load Packages

15.2 Approaches to Prediction

There are two primary approaches to prediction: human judgment and the actuarial (i.e., statistical) method.

15.2.1 Human Judgment

Using the judgment method of prediction, all gathered information is collected and formulated into a prediction in the person's mind. The person selects, measures, and combines information and produces projections solely according to their experience and judgment. For instance, a proclaimed "fantasy expert" might use their experience, expertise, and judgment to make a prediction about how each player will perform by using whatever information and data they deem to be important, aggregating all of this information in their mind to make the prediction for each player. Professional scouts and coaches use judgment when making predictions or selecting players based on their impressions of the players (Den Hartigh et al., 2018). As an example in popular media, in the movie, "Trouble with the Curve", a professional scout makes a judgment about how well a baseball hitter will do in the major leagues from his impressions of the hitter's ability based on the sound of the ball off the player's bat.

15.2.2 Actuarial/Statistical Method

In the actuarial or statistical method of prediction, information is gathered and combined systematically in an evidence-based statistical prediction formula. The method is based on equations and data, so both are needed.

An example of a statistical method of prediction is the Violence Risk Appraisal Guide (Rice et al., 2013). The Violence Risk Appraisal Guide is used in an attempt to predict violence and is used for parole decisions. For instance, the equation might be something like Equation 15.1:

$$\text{violence risk} = \beta \cdot \text{conduct disorder} + \beta \cdot \text{substance use} + \beta \cdot \text{suspended from school} + \beta \cdot \text{childhood aggression} + \ldots \tag{15.1}$$

Then, based on their score and the established cutoffs, a person is given a "low risk", "medium risk", or "high risk" designation.

An actuarial formula for projecting a Running Back's rushing yards might be something like Equation 15.2:

$$\text{rushing yards} = \beta \cdot \text{rushing yards last season} + \beta \cdot \text{age} + \beta \cdot \text{injury history} + \beta \cdot \text{strength of offensive line} + \ldots \tag{15.2}$$

The beta weights in the actuarial model reflect the relative weight to assign each predictor. For instance, in predicting rushing yards, a player's historical performance is likely the strongest predictor, whereas injury history might be a relatively weaker predictor. Thus, we might give historical performance a beta of 3 and injury history a beta of 1 to give a player's historical performance three times more weight than the player's injury history in predicting their rushing yards. For generating the actuarial model, you could obtain the beta weights for each predictor from multiple regression, from machine learning, or from prior research on the relative importance of each predictor.

As an example of using the actuarial approach, Billy Beane, who was the general manager of the Oakland Athletics at the time, wanted to find ways for his team—which had less finanical resources than its competitors—to compete with teams that hard more money to sign players. Because the team did not have the resources to sign the best players, they had to find to find other ways to find the optimal players that they could afford. So, he used statistical formulas that weight variables, such as on-base percentage and slugging percentage, according to their value for winning games, for the use of selecting players. His approach became well-known based on the Michael Lewis book, "Moneyball: The Art of Winning an Unfair Game", and the eventual movie, "Moneyball".

15.2.3 Combining Human Judgment and Statistical Algorithms

There are numerous ways in which humans and statistical algorithms could be involved. On one extreme, humans make all judgments. On the other extreme, although humans may be involved in data collection, a statistical formula makes all decisions based on the input data, consistent with an actuarial approach. However, the human judgment and actuarial approaches can be combined in a hybrid way (Dana & Thomas, 2006). For example, to save time and money, a clinical psychologist might use an actuarial approach in all cases, but might only use a judgment approach when the actuarial approach gives a "positive" test. Or, the clinical psychologist might use both human judgment and an actuarial approach independently to see whether they agree. That is, the clinician may make a prediction based on their judgment and might also generate a prediction from an actuarial approach.

The challenge is what to do when the human and the algorithm disagree. Hypothetically, humans reviewing and adjusting the results from the statistical algorithm could lead to more accurate prediction. However, human input also could lead to the possibility or exacerbation of biased predictions. In general, with very few exceptions, actuarial approaches are as accurate or more accurate than "expert" judgment (Ægisdóttir et al., 2006; Baird & Wagner, 2000; Dawes et al., 1989; Grove et al., 2000; Grove & Meehl, 1996). This is also likely true with respect to predicting player performance in sports (Den Hartigh et al., 2018). Moreover, the superiority of actuarial approaches to human judgment tends to hold even when the expert is given more information than the actuarial approach (Dawes et al., 1989). Allowing experts to override actuarial predictions consistently leads to lower predictive accuracy (Garb & Wood, 2019).

There is sometimes a misconception that formulas cannot account for qualitative information. However, that is not true. Qualitative information can be scored or coded to be quantified so that it can be included in statistical formulas. For instance, if an expert scout is able to meaningfully assess a player's cognitive and motivational factors (i.e., the "X factor" or "intangibles"), the scout can score this across multiple players and include these data in the actuarial prediction formula. For instance, the scout could use a rating scale (e.g., 1 = "poor"; 2 = "fair"; 3 = "good"; 4 = "very good"; 5 = "excellent") to code (i.e., translate) their qualitative judgment into a quantifiable rating that can be integrated with other information in the actuarial formula. That said, the quality of predictions rests on the quality and relevance of the assessment information for the particular prediction decision. If the assessment data are lousy, it is unlikely that a statistical algorithm (or a human for that matter) will make an accurate prediction: "Garbage in, garbage out". A statistical formula cannot rescue inaccurate assessment data.

15.3 Errors in Human Judgment

Human judgment is naturally subject to errors. Common heuristics, cognitive biases, and fallacies[1] are described in Chapter 14. Below, I describe a few errors to which human judgment seems particularly prone.

When operating freely, clinicians and medical experts (and humans more generally) tend to overestimate exceptions to the established rules (i.e., the broken leg syndrome). Meehl (1957) acknowledged that there may be some situations where it is glaringly obvious that the statistical formula would be incorrect because it fails to account for an important factor. He called these special cases "broken leg" cases, in which the human should deviate from the formula (i.e., broken leg countervailing). The example goes like this:

> "If a sociologist were predicting whether Professor X would go to the movies on a certain night, he might have an equation involving age, academic specialty, and introversion score. The equation might yield a probability of .90 that Professor X goes to the movie tonight. But if the family doctor announced that Professor X had just broken his leg, no sensible sociologist would stick with the equation. Why didn't the factor of 'broken leg' appear in the formula? Because broken legs are very rare, and in the sociologist's entire sample of 500 criterion cases plus 250 cross-validating cases, he did not come upon a single instance of it. He uses the broken leg datum confidently, because 'broken leg' is a subclass of a larger class we may crudely denote as 'relatively immobilizing illness or injury,' and movie-attending is a subclass of a larger class of 'actions requiring moderate mobility.' " (Meehl, 1957, pp. 269–270)

However, people too often think that cases where they disagree with the statistical algorithm are broken leg cases. People too often think their case is an exception to the rule. As a result, they too often change the result of the statistical algorithm and are more likely to be wrong than right in doing so. Because actuarial methods are based on actual population levels (i.e., base rates), unique exceptions are not overestimated.

[1] sec-fallacies

Actuarial predictions are perfectly reliable—they will always return the same conclusion given an identical set of data. The human judge is likely to both disagree with others and with themselves given the same set of symptoms.

The decision by an expert (all by all humans) is likely to be influenced by past experiences. Actuarial methods are based on objective algorithms, and past personal experience and personal biases do not factor into any decisions. Humans give weight to less relevant information, and often give too much weight to singular variables. Actuarial formulas do a better job of focusing on relevant variables. Computers are good at factoring in base rates. Humans ignore base rates (base rate neglect).

Computers are better at accurately weighing predictors and calculating unbiased risk estimates. In an actuarial formula, the relevant predictors are weighted according to their predictive power.

Humans are typically given no feedback on their judgments. To improve accuracy of judgments, it is important for feedback to be clear, consistent, and timely.

15.4 Humans Versus Computers

15.4.1 Advantages of Computers

Here are some advantages of computers over humans:

- Computers can process lots of information simultaneously. So can humans. But computers can to an even greater degree.
- Computers are faster at making calculations.
- Computations by computers are error-free (as long as the computations are programmed correctly).
- Computers' judgments will not be biased by fatigue or emotional responses.
- Computers' judgments will tend not to be biased in the way that humans' cognitive biases are. Computers are less likely to be overconfident in their judgments.
- Computers can more accurately weight the set of predictors based on large data sets. Humans tend to give too much weight to singular predictors.

15.4.2 Advantages of Humans

Computers are bad at some things too. Here are some advantages of humans over computers (as of now):

- Humans can be better at identifying patterns in data (but also can mistakenly identify patterns where there are none—i.e., illusory correlation).
- Humans can be flexible and take a different approach if a given approach is not working.
- Humans are better at tasks requiring creativity and imagination, such as developing theories that explain phenomena.
- Humans have the ability to reason, which is especially important when dealing with complex, abstract, or open-ended problems, or problems that have not been faced before (or for which we have insufficient data).
- Humans are better able to learn.
- Humans are better at holistic, gestalt processing, including facial and linguistic processing.

There *may* be situations in which a human judgment would do better than an actuarial judgment. One situation where human judgment would be important is when no actuarial method exists for the judgment or prediction. For instance, when no actuarial method exists for the diagnosis or disorder (e.g., suicide), it is up to the clinician. However, we could collect data on the outcomes or on clinicians' judgments to develop an actuarial method that will be more reliable than the clinicians' judgments. That is, an actuarial method developed based on clinicians' judgments will be more accurate than clinicians' judgments. In other words, we do not necessarily need outcome data to develop an actuarial method. We could use the client's data as predictors of the clinicians' judgments to develop a structured approach to prediction that weighs factors similarly to clinicians, but with more reliable predictions.

Another situation in which human judgment could outperform a statistical algorithm is in true "broken leg" cases, e.g., important and rare events (edge cases) that are not yet accounted for by the algorithm.

Another situation in which human judgment could be preferable is if advanced, complex theories exist. Computers have a difficult time adhering to complex theories, so clinicians may be better suited. However, we do not have any of these complex theories in psychology that are accurate. We would need strong theory informed by data regarding causal influences, and accurate measures to assess them. However, no theories in psychology are that good. Nevertheless, predictive accuracy can be improved when considering theory (Garb & Wood, 2019; Silver, 2012).

If the prediction requires complex configural relations that a computer will have a difficult time replicating, a clinician's judgment may be preferred. Although the likelihood that a person can accurately work through these complex relations is theoretically possible, it is highly unlikely. Holistic pattern recognition (such as language and faces) tends to be better by humans than computers. But computers are getting better with holistic pattern recognition through machine learning.

In sum, the human seeks to integrate information to make a decision, but is biased.

15.4.3 Comparison of Evidence

Hundreds of studies have examined clinical versus actuarial prediction methods across many disciplines. Findings consistently show that actuarial methods are as accurate or more accurate than human judgment/prediction methods. "There is no controversy in social science that shows such a large body of qualitatively diverse studies coming out so uniformly…as this one" (Meehl, 1986, pp. 373–374).

Actuarial methods are particularly valuable for criterion-referenced assessment tasks, in which the aim is to predict specific events or outcomes (Garb & Wood, 2019). For instance, actuarial methods have shown promise in predicting violence, criminal recidivism, psychosis onset, course of mental disorders, treatment selection, treatment failure, suicide attempts, and suicide (Garb & Wood, 2019).

Moreover, actuarial methods are explicit; they can be transparent and lead to informed scientific criticism to improve them. By contrast, human judgment methods are not typically transparent; human judgment relies on mental processes that are often difficult to specify.

15.5 Why Judgment is More Widely Used Than Statistical Formulas

Despite actuarial methods being generally more accurate than human judgment, judgment is much more widely used by clinicians. There are several reasons why actuarial methods have not caught on; one reason is professional traditions. Experts in any field do not like to think that a computer could outperform them. Some practitioners argue that judgment/prediction is an "art form" and that using a statistical formula is treating people like a number. However, using an approach (i.e., human judgment) that systematically leads to less accurate decisions and predictions is an ethical problem.

Some clinicians do not think that group averages (e.g., in terms of which treatment is most effective) apply to an individual client. This invokes the distinction between nomothetic (group-level) inferences and idiographic (individual-level) inferences. However, the scientific evidence and probability theory strongly indicate that it is better to generalize from group-level evidence than throwing out all the evidence and taking the approach of "anything goes."

Clinicians frequently believe the broken leg fallacy, i.e., thinking that your client is an exception to the algorithmic prediction. In most cases, deviating from the statistical formula will result in less accurate predictions. People tend to overestimate the probability of low base rate conditions and events.

Another reason why actuarial methods have not caught on is the belief that receiving a treatment is the only thing that matters. But it is an empirical question which treatment is most effective for whom. What if we could do better? For example, we could potentially use a formula to identify the most effective treatment for a client. Some treatments are no better than placebo; other treatments are actually harmful (Lilienfeld, 2007; Williams et al., 2021).

Another reason why judgment methods are more widely used than actuarial methods is that so-called "experts" (and people in general) show overconfidence in their predictions—clinicians, experts, and humans in general think they are more accurate than they actually are. We see this when examining their calibration; their predictions tend to be miscalibrated. For example, things they report with 80% confidence occur less than 80% of the time, an example of overprecision in their predictions. Humans will sometimes be correct by chance, and they tend to mis-attribute that to their skill; humans tend to remember the successes and forget the failures.

Another argument against using actuarial methods is that "no methods exist". In some cases, that is true—actuarial methods do not yet exist for some prediction problems. However, one can always create an algorithm of the experts' judgments, even if one does not have access to the outcome information. A model of clinicians' responses tends to be more accurate than clinicians' judgments themselves because the model gives the same outcome with the same input data—i.e., it is perfectly reliable.

Another argument from some clinicians is that, "My job is to understand, not to predict". But what kind of understanding does not involve predictions? Accurate predictions help in understanding. Knowing how people would perform in different conditions is the same thing as good understanding.

15.6 Steps to Conduct Actuarial Approaches

Here are several steps to conduct actuarial approaches (Den Hartigh et al., 2018; Kahneman, 2011):

1. Determine a set of relevant variables to measure
2. Determine how you will combine the variables
    - Do some variables have more weight than other variables?

3. Determine how the variables will be scored
    - e.g., a 7-point likert scale
4. Combine the scores based on the pre-specified formula
5. Use the final score to make your prediction (for selecting players)

15.7 Challenges of Data-Driven Approaches and How to Address

There are various challenges of data-driven approaches. First, they are sometimes not interpretable or consistent with theory. Second, they tend to overfit the data. Overfitting is described in Section 11.6. Third, as a result of overfitting the data, they tend to show shrinkage.

15.7.1 Shrinkage

In general, there is often shrinkage of estimates from training data set to a test data set. *Shrinkage* is when variables with stronger predictive power in the original data set tend to show somewhat smaller predictive power (smaller regression coefficients) when applied to new groups. Shrinkage reflects a model overfitting—i.e., when the model explains error variance by capitalizing on chance. Shrinkage is especially likely when the original sample is small and/or unrepresentative and the number of variables considered for inclusion is large. To help minimize the extent of shrinkage, it is recommended to apply cross-validation.

15.7.2 Cross-Validation

Cross-validation with large, representative samples can help evaluate the amount of shrinkage of estimates, particularly for more complex models such as machine learning models (Ursenbach et al., 2019). Ideally, cross-validation would be conducted with a separate sample (external cross-validation) to see the generalizability of estimates. However, you can also do internal cross-validation. For example, you can perform k-fold cross-validation, where you:

- split the data set into k groups
- for each unique group:
    - take the group as a hold-out data set (also called a test data set)

- take the remaining groups as a training data set
- fit a model on the training data set and evaluate it on the test data set
- after all k-folds have been used as the test data set, and all models have been fit, you average the estimates across the models, which presumably yields more robust, generalizable estimates

15.8 Best Actuarial Approaches to Prediction

The best actuarial models tend to be relatively simple (parsimonious), that can account for one or several of the most important predictors and their optimal weightings, and that account for the base rate of the phenomenon. Even unit-weighted formulas (formulas whose predictor variables are equally weighted with a weight of one) can sometimes generalize better to other samples than complex weightings (Garb & Wood, 2019). Differential weightings sometimes capture random variance and over-fit the model, thus leading to predictive accuracy shrinkage in cross-validation samples (Garb & Wood, 2019), as described below. The choice of predictor variables often matters more than their weighting.

An emerging technique that holds promise for increasing predictive accuracy of actuarial methods is machine learning (Garb & Wood, 2019). However, one challenge of some machine learning techniques is that they are like a "black box" and are not transparent, which raises ethical concerns (Garb & Wood, 2019). Moreover, machine learning also tends to lead to overfitting and shrinkage. machine learning may be most valuable when the data available are complex and there are many predictor variables (Garb & Wood, 2019), and when the model is validated with cross-validation.

15.9 Conclusion

In general, it is better to develop and use structured, actuarial approaches than informal approaches that rely on human judgment or judgment by "so-called" experts. Actuarial approaches to prediction tend to be as accurate or more accurate than expert judgment. Nevertheless, in many domains, human judgment tends to be much more widely used than actuarial approaches.

16

Base Rates

16.1 Getting Started

16.1.1 Load Packages

```
library("petersenlab")
```

16.2 Overview

Predicting player performance is a complex prediction task. Performance is probabilistically influenced by many processes, including processes internal to the player in addition to external processes. Moreover, people's performance occurs in the context of a dynamic system with nonlinear, probabilistic, and cascading influences that change across time. The ever-changing system makes behavior challenging to predict. And, similar to chaos theory, one small change in the system can lead to large differences later on. Moreover, there are important factors to keep in mind when making predictions.

Let's consider a prediction example, assuming the following probabilities:

- The probability of contracting HIV is .3%
- The probability of a positive test for HIV is 1%
- The probability of a positive test if you have HIV is 95%

What is the probability of HIV if you have a positive test?

As we will see, the probability is: $\frac{95\% \times .3\%}{1\%} = 28.5\%$. So based on the above probabilities, if you have a positive test, the probability that you have HIV is 28.5%. Most people tend to vastly overestimate the likelihood that the person has HIV in this example. Why? Because they do not pay enough attention to the base rate (in this example, the base rate of HIV is .3%).

16.3 Issues Around Probability

16.3.1 Types of Probabilities

It is important to distinguish between different types of probabilities: marginal probabilities, joint probabilities, and conditional probabilities.

16.3.1.1 Base Rate (Marginal Probability)

The *base rate* is a marginal probability, which is the general probability of an event irrespective of other things. For instance, the base rate of HIV is the probability of developing HIV. In the U.S., the prevalence rate of HIV is ~0.4% of the adult population[1] (archived at https://perma.cc/8GE6-GAPC).

For instance, we can consider the following marginal probabilities:

$P(C_i)$ is the probability (i.e., base rate) of a classification, C, independent of other things. A base rate is often used as the "*prior probability*" in a Bayesian model. In our example above, $P(C_i)$ is the base rate (i.e., prevalence) of HIV in the population: $P(\text{HIV}) = .3\%$. $P(R_i)$ is the probability (base rate) of a response, R, independent of other things. In the example above, $P(R_i)$ is the base rate of a positive test for HIV: $P(\text{positive test}) = 1\%$. The base rate of a positive test is known as the *positivity rate* or *selection ratio*.

16.3.1.2 Joint Probability

A *joint probability* is the probability of two (or more) events occurring simultaneously. For instance, the probability of events A and B both occurring together is $P(A, B)$. A joint probability can be calculated using the marginal probability of each event, as in Equation 16.1:

$$P(A, B) = P(A) \cdot P(B) \tag{16.1}$$

Conversely (and rearranging the terms for the calculation of conditional probability), a joint probability can also be calculated using the conditional probability and marginal probability, as in Equation 16.2:

$$P(A, B) = P(A|B) \cdot P(B) \tag{16.2}$$

[1] https://map.aidsvu.org/profiles/nation/usa/overview

16.3.1.3 Conditional Probability

A *conditional probability* is the probability of one event occurring given the occurrence of another event. Conditional probabilities are written as: $P(A|B)$. This is read as the probability that event A occurs given that event B occurred. For instance, we can consider the following conditional probabilities:

$P(C|R)$ is the probability of a classification, C, given a response, R. In other words, $P(C|R)$ is the probability of having HIV given a positive test: $P(\text{HIV}|\text{positive test})$. $P(R|C)$ is the probability of a response, R, given a classification, C. In the example above, $P(R|C)$ is the probability of having a positive test given that a person has HIV: $P(\text{positive test}|\text{HIV}) = 95\%$.

A conditional probability can be calculated using the joint probability and marginal probability (base rate), as in Equation 16.3:

$$P(A, B) = P(A|B) \cdot P(B) \tag{16.3}$$

16.3.2 Confusion of the Inverse

A conditional probability is not the same thing as its reverse (or inverse) conditional probability. Unless the base rate of the two events (C and R) are the same, $P(C|R) \neq P(R|C)$. However, people frequently make the mistake of thinking that two inverse conditional probabilities are the same. This mistake is known as the "confusion of the inverse", or the "inverse fallacy", or the "conditional probability fallacy". The confusion of inverse probabilities is the logical error of representative thinking that leads people to assume that the probability of C given R is the same as the probability of R given C, even though this is not true. As a few examples to demonstrate the logical fallacy, if 93% of breast cancers occur in high-risk women, this does not mean that 93% of high-risk women will eventually get breast cancer. As another example, if 77% of car accidents take place within 15 miles of a driver's home, this does not mean that you will get in an accident 77% of times you drive within 15 miles of your home.

Which car is the most frequently stolen? It is often the Honda Accord or Honda Civic—probably because they are among the most popular/commonly available cars. The probability that the car is a Honda Accord given that a car was stolen (p(Honda Accord | Stolen)) is what the media reports and what the police care about. However, that is not what buyers and car insurance companies should care about. Instead, they care about the probability that the car will be stolen given that it is a Honda Accord (p(Stolen | Honda Accord)).

Applied to fantasy football, the probability that a given player will be injured given that he is a Running Back (p(Injured | RB)) is not the same as the

probability that a given player is a Running Back given that he is injured (p(RB | Injured)).

16.3.3 Bayes' Theorem

An alternative way of calculating a conditional probability is using the inverse conditional probability (instead of the joint probability). This is known as Bayes' theorem. Bayes' theorem can help us calculate a conditional probability of some classification, C, given some response, R, if we know the inverse conditional probability and the base rate (marginal probability) of each. Bayes' theorem is in Equation 16.4:

$$P(C|R) = \frac{P(R|C) \cdot P(C_i)}{P(R_i)} \tag{16.4}$$

Or, equivalently (rearranging the terms):

$$\frac{P(C|R)}{P(R|C)} = \frac{P(C_i)}{P(R_i)} \tag{16.5}$$

Or, equivalently (rearranging the terms):

$$\frac{P(C|R)}{P(C_i)} = \frac{P(R|C)}{P(R_i)} \tag{16.6}$$

More generally, Bayes' theorem has been described as:

$$\begin{aligned} P(H|E) &= \frac{P(E|H) \cdot P(H)}{P(E)} \\ \text{posterior probability} &= \frac{\text{likelihood} \times \text{prior probability}}{\text{model evidence}} \end{aligned} \tag{16.7}$$

where H is the hypothesis, and E is the evidence—the new information that was not used in computing the prior probability.

In Bayesian terms, the *posterior probability* is the conditional probability of one event occurring given another event—it is the updated probability after the evidence is considered. In this case, the posterior probability is the probability of the classification occurring (C) given the response (R). The *likelihood* is the inverse conditional probability—the probability of the response (R) occurring given the classification (C). The *prior probability* is the marginal probability of the event (i.e., the classification) occurring, before we take into account any new information. The *model evidence* is the marginal probability of the other event occurring—i.e., the marginal probability of seeing the evidence.

Bayes' theorem provides the foundation for a paradigm of statistics called Bayesian statistics, which (unlike frequentist statistics) does not use p-values.

In the HIV example above, we can calculate the conditional probability of HIV given a positive test using three terms: the conditional probability of a positive test given HIV (i.e., the sensitivity of the test), the base rate of HIV, and the base rate of a positive test for HIV. The conditional probability of HIV given a positive test is in Equation 16.8:

$$
\begin{aligned}
P(C|R) &= \frac{P(R|C) \cdot P(C_i)}{P(R_i)} \\
P(\text{HIV}|\text{positive test}) &= \frac{P(\text{positive test}|\text{HIV}) \cdot P(\text{HIV})}{P(\text{positive test})} \\
&= \frac{\text{sensitivity of test} \times \text{base rate of HIV}}{\text{base rate of positive test}} \\
&= \frac{95\% \times .3\%}{1\%} = \frac{.95 \times .003}{.01} \\
&= 28.5\%
\end{aligned}
\tag{16.8}
$$

The `petersenlab`[2] package (Petersen, 2024a) contains the `pAgivenB()` function that estimates the probability of one event, A, given another event, B.

```
petersenlab::pAgivenB(
  pBgivenA = .95,
  pA = .003,
  pB = .01)
```

```
[1] 0.285
```

Thus, assuming the probabilities in the example above, the conditional probability of having HIV if a person has a positive test is 28.5%. Given a positive test, chances are higher than not that the person does not have HIV.

Now let's see what happens if the person tests positive a second time. We would revise our "prior probability" for HIV from the general prevalence in the population (0.3%) to be the "posterior probability" of HIV given a first positive test (28.5%). This is known as *Bayesian updating*. We would also update the "evidence" to be the marginal probability of getting a second positive test.

If we do not know a marginal probability (i.e., base rate) of an event (e.g., getting a second positive test), we can calculate a marginal probability with the *law of total probability* using conditional probabilities and the marginal

[2] https://cran.r-project.org/web/packages/petersenlab/index.html

probability of another event (e.g., having HIV). According to the law of total probability, the probability of getting a positive test is the probability that a person with HIV gets a positive test (i.e., sensitivity) times the base rate of HIV plus the probability that a person without HIV gets a positive test (i.e., false positive rate) times the base rate of not having HIV, as in Equation 16.9:

$$\begin{aligned} P(\text{not } C_i) &= 1 - P(C_i) \\ P(R_i) &= P(R|C) \cdot P(C_i) + P(R|\text{not } C) \cdot P(\text{not } C_i) \\ 1\% &= 95\% \times .3\% + P(R|\text{not } C) \times 99.7\% \end{aligned} \tag{16.9}$$

In this case, we know the marginal probability ($P(R_i)$), and we can use that to solve for the unknown conditional probability that reflects the false positive rate ($P(R|\text{not } C)$), as in Equation 16.10:

$$\begin{aligned} P(R_i) &= P(R|C) \cdot P(C_i) + P(R|\text{not } C) \cdot P(\text{not } C_i) \\ P(R_i) - [P(R|\text{not } C) \cdot P(\text{not } C_i)] &= P(R|C) \cdot P(C_i) && \text{Move } P(R|\text{not } C) \text{ to the left side} \\ -[P(R|\text{not } C) \cdot P(\text{not } C_i)] &= P(R|C) \cdot P(C_i) - P(R_i) && \text{Move } P(R_i) \text{ to the right side} \\ P(R|\text{not } C) \cdot P(\text{not } C_i) &= P(R_i) - [P(R|C) \cdot P(C_i)] && \text{Multiply by } -1 \\ P(R|\text{not } C) &= \frac{P(R_i) - [P(R|C) \cdot P(C_i)]}{P(\text{not } C_i)} && \text{Divide by } P(R|\text{not } C) \\ &= \frac{1\% - [95\% \times .3\%]}{99.7\%} = \frac{.01 - [.95 \times .003]}{.997} \\ &= .7171515\% \end{aligned} \tag{16.10}$$

The `petersenlab`[3] package (Petersen, 2024a) contains the `pBgivenNotA()` function that estimates the probability of one event, B, given that another event, A, did not occur.

```
petersenlab::pBgivenNotA(
  pBgivenA = .95,
  pA = .003,
  pB = .01)
```

```
[1] 0.007171515
```

With this conditional probability ($P(R|\text{not } C)$), the updated marginal probability of having HIV ($P(C_i)$), and the updated marginal probability of not having HIV ($P(\text{not } C_i)$), we can now calculate an updated estimate of the marginal probability of getting a second positive test. The probability of getting a second positive test is the probability that a person with HIV gets a second positive test (i.e., sensitivity) times the updated probability of HIV plus the probability that a person without HIV gets a second positive test (i.e., false positive rate) times the updated probability of not having HIV, as in Equation 16.11:

[3] https://cran.r-project.org/web/packages/petersenlab/index.html

$$
\begin{aligned}
P(R_i) &= P(R|C) \cdot P(C_i) + P(R|\text{not } C) \cdot P(\text{not } C_i) \\
&= 95\% \times 28.5\% + .7171515\% \times 71.5\% = .95 \times .285 + .007171515 \times .715 \\
&= 27.58776\%
\end{aligned}
\tag{16.11}
$$

The `petersenlab`[4] package (Petersen, 2024a) contains the `pB()` function that estimates the marginal probability of one event, B.

```
petersenlab::pB(
  pBgivenA = .95,
  pA = .285,
  pBgivenNotA = .007171515)
```

```
[1] 0.2758776
```

We then substitute the updated marginal probability of HIV ($P(C_i)$) and the updated marginal probability of getting a second positive test ($P(R_i)$) into Bayes' theorem to get the probability that the person has HIV if they have a second positive test (assuming the errors of each test are independent, i.e., uncorrelated), as in Equation 16.12:

$$
\begin{aligned}
P(C|R) &= \frac{P(R|C) \cdot P(C_i)}{P(R_i)} \\
P(\text{HIV}|\text{a second positive test}) &= \frac{P(\text{a second positive test}|\text{HIV}) \cdot P(\text{HIV})}{P(\text{a second positive test})} \\
&= \frac{\text{sensitivity of test} \times \text{updated base rate of HIV}}{\text{updated base rate of positive test}} \\
&= \frac{95\% \times 28.5\%}{27.58776\%} \\
&= 98.14\%
\end{aligned}
\tag{16.12}
$$

The `petersenlab`[5] package (Petersen, 2024a) contains the `pAgivenB()` function that estimates the probability of one event, A, given another event, B.

```
petersenlab::pAgivenB(
  pBgivenA = .95,
  pA = .285,
  pB = .2758776)
```

```
[1] 0.9814135
```

[4] https://cran.r-project.org/web/packages/petersenlab/index.html
[5] https://github.com/DevPsyLab/petersenlab

Thus, a second positive test greatly increases the posterior probability that the person has HIV from 28.5% to over 98%.

As seen in the rearranged formula in Equation 16.5, the ratio of the conditional probabilities is equal to the ratio of the base rates. Thus, it is important to consider base rates. People have a strong tendency to ignore (or give insufficient weight to) base rates when making predictions. The failure to consider the base rate when making predictions when given specific information about a case is known as the base rate fallacy or as base rate neglect. For example, people tend to say that the probability of a rare event is more likely than it actually is given specific information.

As seen in the rearranged formula in Equation 16.6, the inverse conditional probabilities ($P(C|R)$ and $P(R|C)$) are not equal unless the base rates of C and R are the same. If the base rates are not equal, we are making at least some prediction errors. If $P(C_i) > P(R_i)$, our predictions must include some false negatives. If $P(R_i) > P(C_i)$, our predictions must include some false positives.

In sum, the marginal probability, including the prior probability or base rate, should be weighed heavily in predictions unless there are sufficient data to indicate otherwise, i.e., to update the posterior probability based on new evidence. Bayes' theorem provides a powerful tool to anchor predictions to the base rate unless sufficient evidence changes the posterior probability (by updating the evidence and prior probability).

16.4 Base Rate of Rookie Performance

16.4.1 Quarterbacks

16.4.2 Running Backs

16.5 How to Account for Base Rates

There are various ways to account for base rates, including the use of actuarial formulas and the use of Bayesian updating.

16.5.1 Actuarial Formula

One approach to account for base rates is to use actuarial formulas (rather than human judgment) to make the predictions. Actuarial formulas based on multiple regression or machine learning can account for the base rate of the event.

16.5.2 Bayesian Updating

Another approach to account for base rates is to leverage Bayes' theorem, using Bayesian updating and the probability nomogram. Bayesian updating is a form of anchoring and adjustment; however, unlike the anchoring and adjustment heuristic, it is a systematic approach to anchoring and adjustment that anchors one's predictions to the base rate, and then adjusts according to new information. That is, we start with a pretest probability (i.e., base rate) and update our predictions based on the extent of new information (i.e., the likelihood ratio).

To perform Bayesian updating involves comparing the relative probability of two outcomes, $P(C|R)$ versus $P(\text{not } C|R)$. If we want to compare the relative probability of two outcomes, we can use the odds form of Bayes' theorem, as in Equation 16.13:

$$\begin{aligned}
P(C|R) &= \frac{P(R|C) \cdot P(C_i)}{P(R_i)} \\
P(\text{not } C|R) &= \frac{P(R|\text{not } C) \cdot P(\text{not } C_i)}{P(R_i)} \\
\frac{P(C|R)}{P(\text{not } C|R)} &= \frac{\frac{P(R|C) \cdot P(C_i)}{P(R_i)}}{\frac{P(R|\text{not } C) \cdot P(\text{not } C_i)}{P(R_i)}} \\
&= \frac{P(R|C) \cdot P(C_i)}{P(R|\text{not } C) \cdot P(\text{not } C_i)} \\
&= \frac{P(C_i)}{P(\text{not } C_i)} \times \frac{P(R|C)}{P(R|\text{not } C)} \\
\text{posterior odds} &= \text{prior odds} \times \text{likelihood ratio}
\end{aligned} \tag{16.13}$$

As presented in Equation 16.13, the posttest (or posterior) odds are equal to the pretest odds multiplied by the likelihood ratio. Below, we describe the likelihood ratio.

16.5.2.1 Diagnostic Likelihood Ratio

A likelihood ratio is the ratio of two probabilities. It can be used to compare the likelihood of two possibilities. The diagnostic likelihood ratio is an index

of the predictive validity of an instrument: it is the ratio of the probability that a test result is correct to the probability that the test result is incorrect. The diagnostic likelihood ratio is also called the risk ratio. There are two types of diagnostic likelihood ratios: the positive likelihood ratio and the negative likelihood ratio.

16.5.2.1.1 Positive Likelihood Ratio (LR+)

The positive likelihood ratio (LR+) compares the true positive rate to the false positive rate. Positive likelihood ratio values range from 1 to infinity. Higher values reflect greater accuracy, because it indicates the degree to which a true positive is more likely than a false positive. The formula for calculating the positive likelihood ratio is in Equation 16.14.

$$\begin{aligned}\text{positive likelihood ratio (LR+)} &= \frac{\text{TPR}}{\text{FPR}} \\ &= \frac{P(R|C)}{P(R|\text{not } C)} \\ &= \frac{P(R|C)}{1 - P(\text{not } R|\text{not } C)} \\ &= \frac{\text{sensitivity}}{1 - \text{specificity}}\end{aligned} \tag{16.14}$$

16.5.2.1.2 Negative Likelihood Ratio (LR−)

The negative likelihood ratio (LR−) compares the false negative rate to the true negative rate. Negative likelihood ratio values range from 0 to 1. Smaller values reflect greater accuracy, because it indicates that a false negative is less likely than a true negative. The formula for calculating the negative likelihood ratio is in Equation 16.15.

$$\begin{aligned}\text{negative likelihood ratio (LR−)} &= \frac{\text{FNR}}{\text{TNR}} \\ &= \frac{P(\text{not } R|C)}{P(\text{not } R|\text{not } C)} \\ &= \frac{1 - P(R|C)}{P(\text{not } R|\text{not } C)} \\ &= \frac{1 - \text{sensitivity}}{\text{specificity}}\end{aligned} \tag{16.15}$$

16.5.2.2 Probability Nomogram

Using Bayes' theorem (described in Section 16.3.3), solving for posttest odds (based on pretest odds and the likelihood ratio, as in Equation 16.13), and converting odds to probabilities, we can use a Fagan probability nomogram to determine the posttest probability following a test result. The calculation of posttest probability is described in INSERT. A *probability nomogram* is a way of visually applying Bayes' theorem to determine the posttest probability of having a condition based on the pretest (or prior) probability and likelihood ratio, as depicted in Figure 16.1. To use a probability nomogram, connect the dots from the starting probability (left line) with the likelihood ratio (middle line) to see the updated probability. The updated (posttest) probability is where the connecting line crosses the third, right line.

Figure 16.1 Probability Nomogram. (Figure retrieved from https://upload.wikimedia.org/wikipedia/commons/thumb/6/66/Fagan_nomogram.svg/945px-Fagan_nomogram.svg.png).

For instance, if the starting probability is 0.5% and the likelihood ratio is 10 (e.g., sensitivity = .90, specificity = .91: likelihood ratio = $\frac{\text{sensitivity}}{1-\text{specificity}} = \frac{.9}{1-.91} = 10$) from a positive test (i.e., positive likelihood ratio), the updated probability is less than 5%, as depicted in Figure 16.2. The `petersenlab`[6] package (Petersen, 2024a) contains the `posttestProbability()` function that

[6] https://github.com/DevPsyLab/petersenlab

estimates the posttest probability of an event, given the pretest probability and the likelihood ratio, or given the pretest probability and the sensitivity (SN) and specificity (SP) of the test.

```
petersenlab::posttestProbability(
  pretestProb = .005,
  likelihoodRatio = 10)
```

```
[1] 0.04784689
```

```
petersenlab::posttestProbability(
  pretestProb = .005,
  SN = .90,
  SP = .91)
```

```
[1] 0.04784689
```

The function can also estimate the posttest probability of an event given the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN):

```
petersenlab::posttestProbability(
  TP = 450,
  TN = 90545,
  FP = 8955,
  FN = 50)
```

```
[1] 0.04784689
```

We discuss true positives (TP), true negatives (TN), false positives (FP), false negatives (FN), sensitivity (SN), and specificity (SP) in INSERT.

If the starting probability is 0.5% and the likelihood ratio is 0.11 from a negative test (i.e., negative likelihood ratio), the updated probability is nearly indistinguishable from zero (0.05%).

```
petersenlab::posttestProbability(
  pretestProb = .005,
  likelihoodRatio = 0.11)
```

```
[1] 0.0005524584
```

Figure 16.2 Probability Nomogram Example. (Figure adapted from https://upload.wikimedia.org/wikipedia/commons/thumb/6/66/Fagan_nomogram.svg/945px-Fagan_nomogram.svg.png. Also provided in: Petersen (2024b) and Petersen (2024c).)

A probability nomogram calculator can be found at the following link: http://araw.mede.uic.edu/cgi-bin/testcalc.pl (archived at https://perma.cc/X8TF-7YBX). The `petersenlab`[7] package (Petersen, 2024a) contains the `nomogrammer()` function that creates a nomogram plot using the positive and negative likelihood ratio or using the sensitivity (SN) and specificity (SP) of the test, as adapted from Adam Chekroud (https://github.com/achekroud/nomogrammer):

```
petersenlab::nomogrammer(
  pretestProb = .005,
  SN = 0.90,
  SP = 0.91)
```

[7] https://github.com/DevPsyLab/petersenlab

```
petersenlab::nomogrammer(
  pretestProb = .005,
  PLR = 10,
  NLR = 0.11)
```

The blue line indicates the posterior probability of the condition given a positive test. The pink line indicates the posterior probability of the condition given a negative test.

The function can also create a nomogram plot using the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN):

```
petersenlab::nomogrammer(
  TP = 450,
  TN = 90545,
  FP = 8955,
  FN = 50)
```

16.6 Conclusion

Fantasy performance—and behavior more generally—is challenging to predict. People commonly demonstrate biases and fallacies when making predictions. People tend to ignore base rates (base rate fallacy) when making predictions. They also tend to confuse inverse conditional probabilities (conditional probability fallacy). Bayes' theorem provides a way to convert from one conditional probability to its inverse conditional probability using the base rate of each event. There are various ways to account for base rates for more accurate predictions, including through the use of actuarial formulas and Bayesian updating. Bayesian updating uses Bayes' theorem to calculate a posttest probability from a pretest probability and a test result (likelihood ratio). The probability nomogram is a visual approach to Bayesian updating.

17

Evaluation of Prediction/Forecasting Accuracy

"Nothing ruins fantasy like reality." – Renee Miller

17.1 Getting Started

17.1.1 Load Packages

```
library("petersenlab")
library("tidyverse")
library("pROC")
library("magrittr")
library("viridis")
```

17.2 Overview

Predictions can come in different types. Some predictions involve categorical data, whereas other predictions involve continuous data. When dealing with a dichotomous (nominal data that are binary) predictor and outcome variable (or continuous data that have been dichotomized using a cutoff), we can

evaluate predictions using a 2x2 table known as a confusion matrix (see INSERT), or with logistic regression models. When dealing with a continuous outcome variable (e.g., ordinal, interval, or ratio data), we can evaluate predictions using multiple regression or similar variants such as structural equation modeling and mixed models.

In fantasy football, we most commonly predict continuous outcome variables (e.g., fantasy points, rushing yards). Nevertheless, it is also important to understand principles in the prediction of categorical outcomes variables.

In any domain, it is important to evaluate the accuracy of predictions, so we can know how (in)accurate we are, and we can strive to continually improve our predictions. Fantasy performance—and human behavior more general—is incredibly challenging to predict. In fantasy football, there is considerable luck/chance/randomness. There are relatively few (i.e. 17) games, and there is a sizeable injury risk for each player in a given game. These and other factors combine to render fantasy football predictions not highly accurate. But, first, let's learn about the various ways we can evaluate the accuracy of predictions.

17.3 Types of Accuracy

There are two primary dimensions of accuracy: (1) discrimination and (2) calibration. Discrimination and calibration are distinct forms of accuracy. Just because predictions are high in one form of accuracy does not mean that they will be high in the other form of accuracy. As described by Lindhiem et al. (2020), predictions can follow any of the following configurations (and anywhere in between):

- high discrimination, high calibration
- high discrimination, low calibration
- low discrimination, high calibration
- low discrimination, low calibration

Some general indexes of accuracy combine discrimination and calibration, as described in Section 17.3.3.

In addition, accuracy indices can be threshold-dependent or -independent and can be scale-dependent or -independent. Threshold-dependent accuracy indices differ based on the cutoff (i.e., threshold), whereas threshold-independent accuracy indices do not. Thus, raising or lowering the cutoff will change threshold-dependent accuracy indices. Scale-dependent accuracy indices depend on the metric/scale of the data, whereas scale-independent accuracy

indices do not. Thus, scale-dependent accuracy indices cannot be directly compared when using measures of differing scales, whereas scale-independent accuracy indices can be compared across data of differing scales.

17.3.1 Discrimination

When dealing with a categorical outcome, discrimination is the ability to separate events from non-events. When dealing with a continuous outcome, discrimination is the strength of the association between the predictor and the outcome. Aspects of discrimination at a particular cutoff (e.g., sensitivity, specificity, area under the ROC curve) are described in INSERT.

17.3.2 Calibration

When dealing with a categorical outcome, calibration is the degree to which a probabilistic estimate of an event reflects the true underlying probability of the event. When dealing with a continuous outcome, calibration is the degree to which the predicted values are close in value to the outcome values. The importance of examining calibration (in addition to discrimination) is described by Lindhiem et al. (2020).

Calibration is relevant to all kinds of predictions, including weather forecasts. For instance, on the days that the meteorologist says there is a 60% chance of rain, it should rain about 60% of the time. Calibration is also important for fantasy football predictions. When projections state that a group of players is each expected to score 200 points, their projections would be miscalibrated if those players scored only 150 points on average.

There are four general patterns of miscalibration: overextremity, underextremity, overprediction, and underprediction (see Figure 17.7). *Overextremity* exists when the predicted probabilites are too close to the extremes (zero or one). *Underextremity* exists when the predicted probabilities are too far away from the extremes. *Overprediction* exists when the predicted probabilities are consistently greater than the observed probabilities. *Underprediction* exists when the predicted probabilities are consistently less than the observed probabilities. For a more thorough description of these types of miscalibration, see Lindhiem et al. (2020).

Indices for evaluating calibration are described in Section 17.7.3.

17.3.3 General Accuracy

General accuracy indices combine estimates of discrimination and calibration.

17.4 Prediction of Categorical Outcomes

To evaluate the accuracy of our predictions for categorical outcome variables (e.g., binary, dichotomous, or nominal data), we can use either threshold-dependent or threshold-independent accuracy indices.

17.5 Prediction of Continuous Outcomes

To evaluate the accuracy of our predictions for continuous outcome variables (e.g., ordinal, interval, or ratio data), the outcome variable does not have cutoffs, so we would use threshold-independent accuracy indices.

17.6 Threshold-Dependent Accuracy Indices

17.6.1 Decision Outcomes

To consider how we can evaluate the accuracy of predictions for a categorical outcome, consider an example adapted from Meehl & Rosen (1955). The military conducts a test of its prospective members to screen out applicants who would likely fail basic training. To evaluate the accuracy of our predictions using the test, we can examine a confusion matrix. A confusion matrix is a matrix that presents the predicted outcome on one dimension and the actual outcome (truth) on the other dimension. If the predictions and outcomes are dichotomous, the confusion matrix is a 2x2 matrix with two rows and two columns that represent four possible predicted-actual combinations (decision outcomes): true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

When discussing the four decision outcomes, "true" means an accurate judgment, whereas "false" means an inaccurate judgment. "Positive" means that the judgment was that the person has the characteristic of interest, whereas "negative" means that the judgment was that the person does not have the characteristic of interest. A *true positive* is a correct judgment (or prediction) where the judgment was that the person has (or will have) the characteristic of interest, and, in truth, they actually have (or will have) the characteristic.

A *true negative* is a correct judgment (or prediction) where the judgment was that the person does not have (or will not have) the characteristic of interest, and, in truth, they actually do not have (or will not have) the characteristic. A *false positive* is an incorrect judgment (or prediction) where the judgment was that the person has (or will have) the characteristic of interest, and, in truth, they actually do not have (or will not have) the characteristic. A *false negative* is an incorrect judgment (or prediction) where the judgment was that the person does not have (or will not have) the characteristic of interest, and, in truth, they actually do have (or will have) the characteristic.

An example of a confusion matrix is in INSERT.

With the information in the confusion matrix, we can calculate the marginal sums and the proportion of people in each cell (in parentheses), as depicted in INSERT.

That is, we can sum across the rows and columns to identify how many people actually showed poor adjustment ($n = 100$) versus good adjustment ($n = 1,900$), and how many people were selected to reject ($n = 508$) versus retain ($n = 1,492$). If we sum the column of predicted marginal sums ($508 + 1,492$) or the row of actual marginal sums ($100 + 1,900$), we get the total number of people ($N = 2,000$).

Based on the marginal sums, we can compute the marginal probabilities, as depicted in INSERT.

The marginal probability of the person having the characteristic of interest (i.e., showing poor adjustment) is called the *base rate* (BR). That is, the base rate is the proportion of people who have the characteristic. It is calculated by dividing the number of people with poor adjustment ($n = 100$) by the total number of people ($N = 2,000$): $BR = \frac{FN+TP}{N}$. Here, the base rate reflects the prevalence of poor adjustment. In this case, the base rate is .05, so there is a 5% chance that an applicant will be poorly adjusted. The marginal probability of good adjustment is equal to 1 minus the base rate of poor adjustment.

The marginal probability of predicting that a person has the characteristic (i.e., rejecting a person) is called the *selection ratio* (SR). The selection ratio is the proportion of people who will be selected (in this case, rejected rather than retained); i.e., the proportion of people who are identified as having the characteristic. The selection ratio is calculated by dividing the number of people selected to reject ($n = 508$) by the total number of people ($N = 2,000$): $SR = \frac{TP+FP}{N}$. In this case, the selection ratio is .25, so 25% of people are rejected. The marginal probability of not selecting someone to reject (i.e., the marginal probability of retaining) is equal to 1 minus the selection ratio.

The selection ratio might be something that the test dictates according to its cutoff score. Or, the selection ratio might be imposed by external factors that place limits on how many people you can assign a positive test value. For instance, when deciding whether to treat a client, the selection ratio may

depend on how many therapists are available and how many cases can be treated.

17.6.2 Percent Accuracy

Based on the confusion matrix, we can calculate the prediction accuracy based on the percent accuracy of the predictions. The percent accuracy is the number of correct predictions divided by the total number of predictions, and multiplied by 100. In the context of a confusion matrix, this is calculated as: $100\% \times \frac{\text{TP}+\text{TN}}{N}$. In this case, our percent accuracy was 78%—that is, 78% of our predictions were accurate, and 22% of our predictions were inaccurate.

17.6.3 Percent Accuracy by Chance

78% sounds pretty accurate. And it is much higher than 50%, so we are doing a pretty good job, right? Well, it is important to compare our accuracy to what accuracy we would expect to get by chance alone, if predictions were made by a random process rather than using a test's scores. Our selection ratio was 25.4%. How accurate would we be if we randomly selected 25.4% of people to reject? To determine what accuracy we could get by chance alone given the selection ratio and the base rate, we can calculate the chance probability of true positives and the chance probability of true negatives. The probability of a given cell in the confusion matrix is a joint probability—the probability of two events occurring simultaneously. To calculate a joint probability, we multiply the probability of each event.

So, to get the chance expectancies of true positives, we would multiply the respective marginal probabilities, as in Equation 17.1:

$$\begin{aligned} P(TP) &= P(\text{Poor adjustment}) \times P(\text{Reject}) \\ &= BR \times SR \\ &= .05 \times .254 \\ &= .0127 \end{aligned} \tag{17.1}$$

To get the chance expectancies of true negatives, we would multiply the respective marginal probabilities, as in Equation 17.2:

$$\begin{aligned} P(TN) &= P(\text{Good adjustment}) \times P(\text{Retain}) \\ &= (1 - BR) \times (1 - SR) \\ &= .95 \times .746 \\ &= .7087 \end{aligned} \tag{17.2}$$

To get the percent accuracy by chance, we sum the chance expectancies for the correct predictions (TP and TN): $.0127 + .7087 = .7214$. Thus, the percent accuracy you can get by chance alone is 72%. This is because most of our predictions are to retain people, and the base rate of poor adjustment is quite low (.05). Our measure with 78% accuracy provides only a 6% increment in correct predictions. Thus, you cannot judge how good your judgment or prediction is until you know how you would do by random chance.

The chance expectancies for each cell of the confusion matrix are in INSERT

17.6.4 Predicting from the Base Rate

Now, let us consider how well you would do if you were to predict from the base rate. Predicting from the base rate is also called "betting from the base rate", and it involves setting the selection ratio by taking advantage of the base rate so that you go with the most likely outcome in every prediction. Because the base rate is quite low (.05), we could predict from the base rate by selecting no one to reject (i.e., setting the selection ratio at zero). Our percent accuracy by chance if we predict from the base rate would be calculated by multiplying the marginal probabilities, as we did above, but with a new selection ratio, as in Equation 17.3:

$$\begin{aligned}
P(TP) &= P(\text{Poor adjustment}) \times P(\text{Reject}) \\
&= BR \times SR \\
&= .05 \times 0 \\
&= 0 \\
P(TN) &= P(\text{Good adjustment}) \times P(\text{Retain}) \\
&= (1 - BR) \times (1 - SR) \\
&= .95 \times 1 \\
&= .95
\end{aligned} \tag{17.3}$$

We sum the chance expectancies for the correct predictions (TP and TN): $0 + .95 = .95$. Thus, our percent accuracy by predicting from the base rate is 95%. This is damning to our measure because it is a much higher accuracy than the accuracy of our measure. That is, we can be much more accurate than our measure simply by predicting from the base rate and selecting no one to reject.

Going with the most likely outcome in every prediction (predicting from the base rate) can be highly accurate (in terms of percent accuracy) as noted by Meehl & Rosen (1955), especially when the base rate is very low or very high. This should serve as an important reminder that we need to compare

the accuracy of our measures to the accuracy by (1) random chance and (2) predicting from the base rate. There are several important implications of the impact of base rates on prediction accuracy. One implication is that using the same test in different settings with different base rates will markedly change the accuracy of the test. Oftentimes, using a test will actually *decrease* the predictive accuracy when the base rate deviates greatly from .50. But percent accuracy is not everything. Percent accuracy treats different kinds of errors as if they are equally important. However, the value we place on different kinds of errors may be different, as described next.

17.6.5 Different Kinds of Errors Have Different Costs

Some errors have a high cost, and some errors have a low cost. Among the four decision outcomes, there are two types of errors: false positives and false negatives. The extent to which false positives and false negatives are costly depends on the prediction problem. So, even though you can often be most accurate by going with the base rate, it may be advantageous to use a screening instrument despite lower overall accuracy because of the huge difference in costs of false positives versus false negatives in some cases.

Consider the example of a screening instrument for HIV. False positives would be cases where we said that someone is at high risk of HIV when they are not, whereas false negatives are cases where we said that someone is not at high risk when they actually are. The costs of false positives include a shortage of blood, some follow-up testing, and potentially some anxiety, but that is about it. The costs of false negatives may be people getting HIV. In this case, the costs of false negatives greatly outweigh the costs of false positives, so we use a screening instrument to try to identify the cases at high risk for HIV because of the important consequences of failing to do so, even though using the screening instrument will lower our overall accuracy level.

Another example is when the Central Intelligence Agency (CIA) used a screen for protective typists during wartime to try to detect spies. False positives would be cases where the CIA believes that a person is a spy when they are not, and the CIA does not hire them. False negatives would be cases where the CIA believes that a person is not a spy when they actually are, and the CIA hires them. In this case, a false positive would be fine, but a false negative would be really bad.

How you weigh the costs of different errors depends considerably on the domain and context. Possible costs of false positives to society include: unnecessary and costly treatment with side effects and sending an innocent person to jail (despite our presumption of innocence in the United States criminal justice system that a person is innocent until proven guilty). Possible costs of false negatives to society include: setting a guilty person free, failing to detect

a bomb or tumor, and preventing someone from getting treatment who needs it.

The differential costs of different errors also depend on how much flexibility you have in the selection ratio in being able to set a stringent versus loose selection ratio. Consider if there is a high cost of getting rid of people during the selection process. For example, if you must hire 100 people and only 100 people apply for the position, you cannot lose people, so you need to hire even high-risk people. However, if you do not need to hire many people, then you can hire more conservatively.

Any time the selection ratio differs from the base rate, you will make errors. For example, if you reject 25% of applicants, and the base rate of poor adjustment is 5%, then you are making errors of over-rejecting (false positives). By contrast, if you reject 1% of applicants and the base rate of poor adjustment is 5%, then you are making errors of under-rejecting or over-accepting (false negatives).

A low base rate makes it harder to make predictions, and tends to lead to less accurate predictions. For instance, it is very challenging to predict low base rate behaviors, including suicide (Kessler et al., 2020). For this reason, it is likely much more challenging to predict touchdowns—which happen relatively less often—than it is to predict passing/rushing/receiving yards—which are more frequent and continuously distributed.

[EVALUATE EMPIRICALLY]

17.6.6 Sensitivity, Specificity, PPV, and NPV

As described earlier, percent accuracy is not the only important aspect of accuracy. Percent accuracy can be misleading because it is highly influenced by base rates. You can have a high percent accuracy by predicting from the base rate and saying that no one has the condition (if the base rate is low) or that everyone has the condition (if the base rate is high). Thus, it is also important to consider other aspects of accuracy, including sensitivity (SN), specificity (SP), positive predictive value (PPV), and negative predictive value (NPV). We want our predictions to be sensitive to be able to detect the characteristic but also to be specific so that we classify only people actually with the characteristic as having the characteristic.

Let us return to the confusion matrix in INSERT. If we know the frequency of each of the four predicted-actual combinations of the confusion matrix (TP, TN, FP, FN), we can calculate sensitivity, specificity, PPV, and NPV.

Sensitivity is the proportion of those with the characteristic (TP + FN) that we identified with our measure (TP): $\frac{\text{TP}}{\text{TP+FN}} = \frac{86}{86+14} = .86$. Specificity is the proportion of those who do not have the characteristic (TN + FP) that we correctly classify as not having the characteristic (TN): $\frac{\text{TN}}{\text{TN+FP}} = \frac{1{,}478}{1{,}478+422} = .78$.

PPV is the proportion of those who we classify as having the characteristic (TP + FP) who actually have the characteristic (TP): $\frac{\text{TP}}{\text{TP+FP}} = \frac{86}{86+422} = .17$. NPV is the proportion of those we classify as not having the characteristic (TN + FN) who actually do not have the characteristic (TN): $\frac{\text{TN}}{\text{TN+FN}} = \frac{1{,}478}{1{,}478+14} = .99$.

Sensitivity, specificity, PPV, and NPV are proportions, and their values therefore range from 0 to 1, where higher values reflect greater accuracy. With sensitivity, specificity, PPV, and NPV, we have a good snapshot of how accurate the measure is at a given cutoff. In our case, our measure is good at finding whom to reject (high sensitivity), but it is rejecting too many people who do not need to be rejected (lower PPV due to many FPs). Most people whom we classify as having the characteristic do not actually have the characteristic. However, the fact that we are over-rejecting could be okay depending on our goals, for instance, if we do not care about over-dropping (i.e., the PPV being low).

17.6.6.1 Some Accuracy Estimates Depend on the Cutoff

Sensitivity, specificity, PPV, and NPV differ based on the cutoff (i.e., threshold) for classification. Consider the following example. Aliens visit Earth, and they develop a test to determine whether a berry is edible or inedible.

Figure 17.1 depicts the distributions of scores by berry type. Note how there are clearly two distinct distributions. However, the distributions overlap to some degree. Thus, any cutoff will have at least some inaccurate classifications. The extent of overlap of the distributions reflects the amount of measurement error of the measure with respect to the characteristic of interest.

```
#No Cutoff
sampleSize <- 1000

edibleScores <- rnorm(sampleSize, 50, 15)
inedibleScores <- rnorm(sampleSize, 100, 15)

edibleData <- data.frame(
  score = c(
    edibleScores,
    inedibleScores),
  type = c(
    rep("edible", sampleSize),
    rep("inedible", sampleSize)))

cutoff <- 75
```

```
hist_edible <- density(
  edibleScores,
  from = 0,
  to = 150) %$% # exposition pipe magrittr::`%$%`
  data.frame(
    x = x,
    y = y) %>%
  mutate(area = x >= cutoff)

hist_edible$type[hist_edible$area == TRUE] <- "edible_FP"
hist_edible$type[hist_edible$area == FALSE] <- "edible_TN"

hist_inedible <- density(
  inedibleScores,
  from = 0,
  to = 150) %$% # exposition pipe magrittr::`%$%`
  data.frame(
    x = x,
    y = y) %>%
  mutate(area = x < cutoff)

hist_inedible$type[hist_inedible$area == TRUE] <- "inedible_FN"
hist_inedible$type[hist_inedible$area == FALSE] <- "inedible_TP"

density_data <- bind_rows(
  hist_edible,
  hist_inedible)

density_data$type <- factor(
  density_data$type,
  levels = c(
    "edible_TN",
    "inedible_TP",
    "edible_FP",
    "inedible_FN"))

ggplot(
  data = edibleData,
  aes(
    x = score,
    ymin = 0,
    fill = type)) +
  geom_density(alpha = .5) +
  scale_fill_manual(
```

```
    name = "Berry Type",
    values = c(
      viridis::viridis(2)[1],
      viridis::viridis(2)[2])) +
  scale_y_continuous(name = "Frequency") +
  theme_bw() +
  theme(
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank())
```

Figure 17.1 Distribution of Test Scores by Berry Type.

Figure 17.2 depicts the distributions of scores by berry type with a cutoff. The red line indicates the cutoff—the level above which berries are classified by the test as inedible. There are errors on each side of the cutoff. Below the cutoff, there are some false negatives (blue): inedible berries that are inaccurately classified as edible. Above the cutoff, there are some false positives (green): edible berries that are inaccurately classified as inedible. Costs of false negatives could include sickness or death from eating the inedible berries. Costs of false positives could include taking longer to find food, finding insufficient food, and starvation.

```
#Standard Cutoff
ggplot(
  data = density_data,
```

```
  aes(
    x = x,
    ymin = 0,
    ymax = y,
    fill = type)) +
  geom_ribbon(alpha = 1) +
  scale_fill_manual(
    name = "Berry Type",
    values = c(
      viridis::viridis(4)[4],
      viridis::viridis(4)[1],
      viridis::viridis(4)[3],
      viridis::viridis(4)[2]),
    breaks = c("edible_TN","inedible_TP","edible_FP","inedible_FN"),
    labels = c("Edible: TN","Inedible: TP","Edible: FP","Inedible: FN")) +
  geom_line(aes(y = y)) +
  geom_vline(
    xintercept = cutoff,
    color = "red",
    linewidth = 2) +
  scale_x_continuous(name = "score") +
  scale_y_continuous(name = "Frequency") +
  theme_bw() +
  theme(
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank())
```

Figure 17.2 Classifications Based on a Cutoff. Note that some true negatives and true positives are hidden behind the false positives and false negatives.

Based on our assessment goals, we might use a different selection ratio by changing the cutoff. Figure 17.3 depicts the distributions of scores by berry type when we raise the cutoff. There are now more false negatives (blue) and fewer false positives (green). If we raise the cutoff (to be more conservative), the number of false negatives increases and the number of false positives decreases. Consequently, as the cutoff increases, sensitivity and NPV decrease (because we have more false negatives), whereas specificity and PPV increase (because we have fewer false positives). A higher cutoff could be optimal if the costs of false positives are considered greater than the costs of false negatives. For instance, if the aliens cannot risk eating the inedible berries because the berries are fatal, and there are sufficient edible berries that can be found to feed the alien colony.

```
#Raise the cutoff
cutoff <- 85

hist_edible <- density(
  edibleScores,
  from = 0,
  to = 150) %$% # exposition pipe magrittr::`%$%`
  data.frame(
    x = x,
    y = y) %>%
```

```
  mutate(area = x >= cutoff)

hist_edible$type[hist_edible$area == TRUE] <- "edible_FP"
hist_edible$type[hist_edible$area == FALSE] <- "edible_TN"

hist_inedible <- density(
  inedibleScores,
  from = 0,
  to = 150) %$% # exposition pipe magrittr::`%$%`
  data.frame(
    x = x,
    y = y) %>%
  mutate(area = x < cutoff)

hist_inedible$type[hist_inedible$area == TRUE] <- "inedible_FN"
hist_inedible$type[hist_inedible$area == FALSE] <- "inedible_TP"

density_data <- bind_rows(
  hist_edible,
  hist_inedible)

density_data$type <- factor(
  density_data$type,
  levels = c(
    "edible_TN",
    "inedible_TP",
    "edible_FP",
    "inedible_FN"))

ggplot(
  data = density_data,
  aes(
    x = x,
    ymin = 0,
    ymax = y,
    fill = type)) +
  geom_ribbon(alpha = 1) +
  scale_fill_manual(
    name = "Berry Type",
    values = c(
      viridis::viridis(4)[4],
      viridis::viridis(4)[1],
      viridis::viridis(4)[3],
      viridis::viridis(4)[2]),
```

```
    breaks = c("edible_TN","inedible_TP","edible_FP","inedible_FN"),
    labels = c("Edible: TN","Inedible: TP","Edible: FP","Inedible: FN")) +
  geom_line(aes(y = y)) +
  geom_vline(
    xintercept = cutoff,
    color = "red",
    linewidth = 2) +
  scale_x_continuous(name = "score") +
  scale_y_continuous(name = "Frequency") +
  theme_bw() +
  theme(
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank())
```

Figure 17.3 Classifications Based on Raising the Cutoff. Note that some true negatives and true positives are hidden behind the false positives and false negatives.

Figure 17.4 depicts the distributions of scores by berry type when we lower the cutoff. There are now fewer false negatives (blue) and more false positives (green). If we lower the cutoff (to be more liberal), the number of false negatives decreases and the number of false positives increases. Consequently, as the cutoff decreases, sensitivity and NPV increase (because we have fewer false negatives), whereas specificity and PPV decrease (because we have more false positives). A lower cutoff could be optimal if the costs of false negatives are

considered greater than the costs of false positives. For instance, if the aliens cannot risk missing edible berries because they are in short supply relative to the size of the alien colony, and eating the inedible berries would, at worst, lead to minor, temporary discomfort.

```
#Lower the cutoff
cutoff <- 65

hist_edible <- density(
  edibleScores,
  from = 0,
  to = 150) %$% # exposition pipe magrittr::`%$%`
  data.frame(
    x = x,
    y = y) %>%
  mutate(area = x >= cutoff)

hist_edible$type[hist_edible$area == TRUE] <- "edible_FP"
hist_edible$type[hist_edible$area == FALSE] <- "edible_TN"

hist_inedible <- density(
  inedibleScores,
  from = 0,
  to = 150) %$% # exposition pipe magrittr::`%$%`
  data.frame(
    x = x,
    y = y) %>%
  mutate(area = x < cutoff)

hist_inedible$type[hist_inedible$area == TRUE] <- "inedible_FN"
hist_inedible$type[hist_inedible$area == FALSE] <- "inedible_TP"

density_data <- bind_rows(
  hist_edible,
  hist_inedible)

density_data$type <- factor(
  density_data$type,
  levels = c(
    "edible_TN",
    "inedible_TP",
    "edible_FP",
    "inedible_FN"))

ggplot(
```

```
  data = density_data,
  aes(
    x = x,
    ymin = 0,
    ymax = y,
    fill = type)) +
  geom_ribbon(alpha = 1) +
  scale_fill_manual(
    name = "Berry Type",
    values = c(
      viridis::viridis(4)[4],
      viridis::viridis(4)[1],
      viridis::viridis(4)[3],
      viridis::viridis(4)[2]),
    breaks = c("edible_TN","inedible_TP","edible_FP","inedible_FN"),
    labels = c("Edible: TN","Inedible: TP","Edible: FP","Inedible: FN")) +
  geom_line(aes(y = y)) +
  geom_vline(
    xintercept = cutoff,
    color = "red",
    linewidth = 2) +
  scale_x_continuous(name = "score") +
  scale_y_continuous(name = "Frequency") +
  theme_bw() +
  theme(
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank())
```

Figure 17.4 Classifications Based on Lowering the Cutoff. Note that some true negatives and true positives are hidden behind the false positives and false negatives.

In sum, sensitivity and specificity differ based on the cutoff for classification. If we raise the cutoff, sensitivity and PPV increase (due to fewer false positives), whereas sensitivity and NPV decrease (due to more false negatives). If we lower the cutoff, sensitivity and NPV increase (due to fewer false negatives), whereas specificity and PPV decrease (due to more false positives). Thus, the optimal cutoff depends on how costly each type of error is: false negatives and false positives. If false negatives are more costly than false positives, we would set a low cutoff. If false positives are more costly than false negatives, we would set a high cutoff.

17.6.7 Signal Detection Theory

Signal detection theory (SDT) is a probability-based theory for the detection of a given stimulus (signal) from a stimulus set that includes non-target stimuli (noise). SDT arose through the development of radar (**RA**dio **D**etection **A**nd **R**anging) and sonar (**SO**und **N**avigation **A**nd **R**anging) in World War II based on research on sensory-perception research. The military wanted to determine which objects on radar/sonar were enemy aircraft/submarines, and which were noise (e.g., different object in the environment or even just the weather itself). SDT allowed determining how many errors operators made (how accurate they were) and decomposing errors into different kinds of er-

rors. SDT distinguishes between sensitivity and bias. In SDT, *sensitivity* (or discriminability) is how well an assessment distinguishes between a target stimulus and non-target stimuli (i.e., how well the assessment detects the target stimulus amid non-target stimuli). *Bias* is the extent to which the probability of a selection decision from the assessment is higher or lower than the true rate of the target stimulus.

Some radar/sonar operators were not as sensitive to the differences between signal and noise, due to factors such as age, ability to distinguish gradations of a signal, etc. People who showed low sensitivity (i.e., who were not as successful at distinguishing between signal and noise) were screened out because the military perceived sensitivity as a skill that was not easily taught. By contrast, other operators could distinguish signal from noise, but their threshold was too low or high—they could take in information, but their decisions tended to be wrong due to systematic bias or poor calibration. That is, they systematically over-rejected or under-rejected stimuli. Over-rejecting leads to many false negatives (i.e., saying that a stimulus is safe when it is not). Under-rejecting leads to many false positives (i.e., saying that a stimulus is harmful when it is not). A person who showed good sensitivity but systematic bias was considered more teach-able than a person who showed low sensitivity. Thus, radar and sonar operators were selected based on their sensitivity to distinguish signal from noise, and then were trained to improve the calibration so they reduce their systematic bias and do not systematically over- or under-reject.

Although SDT was originally developed for use in World War II, it now plays an important role in many areas of science and medicine. A medical application of SDT is tumor detection in radiology. Another application of SDT in society is using x-ray to detect bombs or other weapons. An example of applying SDT to fantasy football could be in the prediction (and evaluation) of whether or not a player scores a touchdown in a game.

SDT metrics of sensitivity include d' ("d-prime"), A (or A'), and the area under the receiver operating characteristic (ROC) curve. SDT metrics of bias include β (beta), c, and b.

17.6.7.1 Receiver Operating Characteristic (ROC) Curve

The x-axis of the ROC curve is the false alarm rate or false positive rate (1− specificity). The y-axis is the hit rate or true positive rate (sensitivity). We can trace the ROC curve as the combination between sensitivity and specificity at every possible cutoff. At a cutoff of zero (top right of ROC curve), we calculate sensitivity (1.0) and specificity (0) and plot it. At a cutoff of zero, the assessment tells us to make an action for every stimulus (i.e., it is the most liberal). We then gradually increase the cutoff, and plot sensitivity and specificity at each cutoff. As the cutoff increases, sensitivity decreases

and specificity increases. We end at the highest possible cutoff, where the sensitivity is 0 and the specificity is 1.0 (i.e., we never make an action; i.e., it is the most conservative). Each point on the ROC curve corresponds to a pair of hit and false alarm rates (sensitivity and specificity) resulting from a specific cutoff value. Then, we can draw lines or a curve to connect the points.

INSERT depicts an empirical ROC plot where lines are drawn to connect the hit and false alarm rates.

INSERT depicts an ROC curve where a smoothed and fitted curve is drawn to connect the hit and false alarm rates.

17.6.7.1.1 Area Under the ROC Curve

ROC methods can be used to compare and compute the discriminative power of measurement devices free from the influence of selection ratios, base rates, and costs and benefits. An ROC analysis yields a quantitative index of how well an index predicts a signal of interest or can discriminate between different signals. ROC analysis can help tell us how often our assessment would be correct. If we randomly pick two observations, and we were right once and wrong once, we were 50% accurate. But this would be a useless measure because it reflects chance responding.

The geometrical area under the ROC curve reflects the discriminative accuracy of the measure. The index is called the **a**rea **u**nder the **c**urve (AUC) of an ROC curve. AUC quantifies the discriminative power of an assessment. AUC is the probability that a randomly selected target and a randomly selected non-target is ranked correctly by the assessment method. AUC values range from 0.0 to 1.0, where chance accuracy is 0.5 as indicated by diagonal line in the ROC curve. That is, a measure can be useful to the extent that its ROC curve is above the diagonal line (i.e., its discriminative accuracy is above chance).

AUC is a threshold-independent accuracy index that applies across all possible cutoff values.

Figure 17.5 depicts ROC curves with a range of AUC values.

```
set.seed(52242)

auc60 <- petersenlab::simulateAUC(.60, 50000)
auc70 <- petersenlab::simulateAUC(.70, 50000)
auc80 <- petersenlab::simulateAUC(.80, 50000)
auc90 <- petersenlab::simulateAUC(.90, 50000)
auc95 <- petersenlab::simulateAUC(.95, 50000)
auc99 <- petersenlab::simulateAUC(.99, 50000)
```

```
plot(
  pROC::roc(
    y ~ x,
    auc60,
    smooth = TRUE),
  legacy.axes = TRUE,
  print.auc = TRUE,
  print.auc.x = .52,
  print.auc.y = .61,
  print.auc.pattern = "%.2f")

plot(
  pROC::roc(
    y ~ x,
    auc70,
    smooth = TRUE),
  legacy.axes = TRUE,
  print.auc = TRUE,
  print.auc.x = .6,
  print.auc.y = .67,
  print.auc.pattern = "%.2f",
  add = TRUE)

plot(
  pROC::roc(
    y ~ x,
    auc80,
    smooth = TRUE),
  legacy.axes = TRUE,
  print.auc = TRUE,
  print.auc.x = .695,
  print.auc.y = .735,
  print.auc.pattern = "%.2f",
  add = TRUE)

plot(
  pROC::roc(
    y ~ x,
    auc90,
    smooth = TRUE),
  legacy.axes = TRUE,
  print.auc = TRUE,
  print.auc.x = .805,
  print.auc.y = .815,
```

```
  print.auc.pattern = "%.2f",
  add = TRUE)

plot(
  pROC::roc(
    y ~ x,
    auc95,
    smooth = TRUE),
  legacy.axes = TRUE,
  print.auc = TRUE,
  print.auc.x = .875,
  print.auc.y = .865,
  print.auc.pattern = "%.2f",
  add = TRUE)

plot(
  pROC::roc(
    y ~ x,
    auc99,
    smooth = TRUE),
  legacy.axes = TRUE,
  print.auc = TRUE,
  print.auc.x = .94,
  print.auc.y = .94,
  print.auc.pattern = "%.2f",
  add = TRUE)
```

Figure 17.5 Receiver Operating Characteristic (ROC) Curves for Various Levels of Area Under The ROC Curve (AUC) for Various Measures.

As an example, given an AUC of .75, this says that the overall score of an individual who has the characteristic in question will be higher 75% of the time than the overall score of an individual who does not have the characteristic. In lay terms, AUC provides the probability that we will classify correctly based on our instrument if we were to randomly pick one good and one bad outcome. AUC is a stronger index of accuracy than percent accuracy, because you can have high percent accuracy just by going with the base rate. AUC tells us how much better than chance a measure is at discriminating outcomes. AUC is useful as a measure of general discriminative accuracy, and it tells us how accurate a measure is at all possible cutoffs. Knowing the accuracy of a measure at all possible cutoffs can be helpful for selecting the optimal cutoff, given the goals of the assessment. In reality, however, we may not be interested in all cutoffs because not all errors are equal in their costs.

If we lower the base rate, we would need a larger sample to get enough people to classify into each group. SDT/ROC methods are traditionally about dichotomous decisions (yes/no), not graded judgments. SDT/ROC methods can get messy with ordinal data that are more graded because you would have an AUC curve for each ordinal grouping.

17.6.8 Accuracy Indices

There are various accuracy indices we can use to evaluate the accuracy of predictions for categorical outcome variables. We have already described several accuracy indices, including percent accuracy, sensitivity, specificity, positive predictive value, negative predictive value, and area under the ROC curve. We describe these and other indices in greater detail below.

The `petersenlab`[1] package (Petersen, 2024a) contains the `accuracyAtCutoff()` function that computes many accuracy indices for the prediction of categorical outcome variables.

```
#petersenlab::accuracyAtCutoff()
```

The `petersenlab`[2] package (Petersen, 2024a) contains the `accuracyAtEachCutoff()` function that computes many accuracy indices for the prediction of categorical outcome variables at each possible cutoff.

```
#petersenlab::accuracyAtEachCutoff()
```

There are also test calculators available online:

- http://araw.mede.uic.edu/cgi-bin/testcalc.pl
- https://dlrs.shinyapps.io/shinyDLRs

17.6.8.1 Confusion Matrix aka 2x2 Accuracy Table aka Cross-Tabulation aka Contingency Table

A confusion matrix (aka 2x2 accuracy table, cross-tabulation table, or contigency table) is a matrix for categorical data that presents the predicted outcome on one dimension and the actual outcome (truth) on the other dimension. If the predictions and outcomes are dichotomous, the confusion matrix is a 2x2 matrix with two rows and two columns that represent four possible predicted-actual combinations (decision outcomes). In such a case, the confusion matrix provides a tabular count of each type of accurate cases (true

[1] https://github.com/DevPsyLab/petersenlab
[2] https://github.com/DevPsyLab/petersenlab

positives and true negatives) versus the number of each type of error (false positives and false negatives), as shown in INSERT. An example of a confusion matrix is in INSERT.

17.6.8.1.1 Number

```
#table(mydata$diagnosisFactor, mydata$diseaseFactor)
```

17.6.8.1.2 Number with margins added

```
#addmargins(table(mydata$diagnosisFactor, mydata$diseaseFactor))
```

17.6.8.1.3 Proportions

```
#prop.table(table(mydata$diagnosisFactor, mydata$diseaseFactor))
```

17.6.8.1.4 Proportions with margins added

```
#addmargins(prop.table(table(mydata$diagnosisFactor, mydata$diseaseFactor)))
```

17.6.8.2 True Positives (TP)

True positives (TPs) are instances in which a positive classification (e.g., stating that a disease is present for a person) is correct—that is, the test says that a classification is present, and the classification is present. True positives are also called valid positives (VPs) or hits. Higher values reflect greater accuracy. The formula for true positives is in Equation 17.4:

$$\text{TP} = \text{BR} \times \text{SR} \times N \tag{17.4}$$

17.6.8.3 True Negatives (TN)

True negatives (TNs) are instances in which a negative classification (e.g., stating that a disease is absent for a person) is correct—that is, the test

says that a classification is not present, and the classification is actually not present. True negatives are also called valid negatives (VNs) or correct rejections. Higher values reflect greater accuracy. The formula for true negatives is in Equation 17.5:

$$\mathrm{TN} = (1 - \mathrm{BR}) \times (1 - \mathrm{SR}) \times N \tag{17.5}$$

17.6.8.4 False Positives (FP)

False positives (FPs) are instances in which a positive classification (e.g., stating that a disease is present for a person) is incorrect—that is, the test says that a classification is present, and the classification is not present. False positives are also called false alarms (FAs). Lower values reflect greater accuracy. The formula for false positives is in Equation Equation 17.6:

$$\mathrm{FP} = (1 - \mathrm{BR}) \times \mathrm{SR} \times N \tag{17.6}$$

17.6.8.5 False Negatives (FN)

False negatives (FNs) are instances in which a negative classification (e.g., stating that a disease is absent for a person) is incorrect—that is, the test says that a classification is not present, and the classification is present. False negatives are also called misses. Lower values reflect greater accuracy. The formula for false negatives is in Equation 17.7:

$$\mathrm{FN} = \mathrm{BR} \times (1 - \mathrm{SR}) \times N \tag{17.7}$$

17.6.8.6 Selection Ratio (SR)

The selection ratio (SR) is the marginal probability of selection, independent of other things: $P(R_i)$. It is not an index of accuracy, per se. In medicine, the selection ratio is the proportion of people who test positive for the disease. In fantasy football, the selection ratio is the proportion of players who you predict will show a given outcome. For example, if you are trying to predict the players who will score a touchdown in a game, the selection ratio is the proportion of players who you predict will score a touchdown. The formula for calculating the selection ratio is in Equation 17.8.

$$\begin{aligned} \mathrm{SR} &= P(R_i) \\ &= \frac{\mathrm{TP} + \mathrm{FP}}{N} \end{aligned} \tag{17.8}$$

17.6.8.7 Base Rate (BR)

The base rate (BR) of a classification is its marginal probability, independent of other things: $P(C_i)$. It is not an index of accuracy, per se. In medicine, the base rate of a disease is its prevalence in the population, as in Equation 17.9. Without additional information, the base rate is used as the initial *pretest probability*. In fantasy football, the base rate is the proportion of players who actually show the particular outcome. For example, if you are trying to predict the players who will score a touchdown in a game, the base rate is the proportion of players who actually score a touchdown in the game. The formula for calculating the selection ratio is in Equation 17.9.

$$\begin{aligned} \text{BR} &= P(C_i) \\ &= \frac{\text{TP} + \text{FN}}{N} \end{aligned} \tag{17.9}$$

17.6.8.8 Pretest Odds

The pretest odds of a classification can be estimated using the pretest probability (i.e., base rate). To convert a probability to odds, divide the probability by one minus that probability, as in Equation 17.10.

$$\text{pretest odds} = \frac{\text{pretest probability}}{1 - \text{pretest probability}} \tag{17.10}$$

17.6.8.9 Percent Accuracy

Percent Accuracy is also called overall accuracy. Higher values reflect greater accuracy. The formula for percent accuracy is in Equation 17.11. Percent accuracy has several problems. First, it treats all errors (FP and FN) as equally important. However, in practice, it is rarely the case that false positives and false negatives are equally important. Second, percent accuracy can be misleading because it is highly influenced by base rates. You can have a high percent accuracy by predicting from the base rate and saying that no one has the characteristic (if the base rate is low) or that everyone has the characteristic (if the base rate is high). Thus, it is also important to consider other aspects of accuracy.

$$\text{Percent Accuracy} = 100\% \times \frac{\text{TP} + \text{TN}}{N} \tag{17.11}$$

17.6.8.10 Percent Accuracy by Chance

The formula for calculating percent accuracy by chance is in Equation 17.12.

$$\begin{aligned}\text{Percent Accuracy by Chance} &= 100\% \times [P(\text{TP}) + P(\text{TN})] \\ &= 100\% \times \{(\text{BR} \times \text{SR}) + [(1 - \text{BR}) \times (1 - \text{SR})]\}\end{aligned} \tag{17.12}$$

17.6.8.11 Percent Accuracy Predicting from the Base Rate

Predicting from the base rate is going with the most likely outcome in every prediction. If the base rate is less than .50, it would involve predicting that the condition is absent for every case. If the base rate is .50 or above, it would involve predicting that the condition is present for every case. Predicting from the base rate is a special case of percent accuracy by chance when the selection ratio is set to either one (if the base rate $\geq .5$) or zero (if the base rate $< .5$).

17.6.8.12 Relative Improvement Over Chance (RIOC)

Relative improvement over chance (RIOC) is a prediction's improvement over chance as a proportion of the maximum possible improvement over chance, as described by Farrington & Loeber (1989). Higher values reflect greater accuracy. The formula for calculating RIOC is in Equation 17.13.

$$\text{relative improvement over chance (RIOC)} = \frac{\text{total correct} - \text{chance correct}}{\text{maximum correct} - \text{chance correct}} \tag{17.13}$$

17.6.8.13 Relative Improvement Over Predicting from the Base Rate

Relative improvement over predicting from the base rate is a prediction's improvement over predicting from the base rate as a proportion of the maximum possible improvement over predicting from the base rate. Higher values reflect greater accuracy. The formula for calculating relative improvement over predicting from the base rate is in Equation 17.14.

$$\text{relative improvement over predicting from base rate} = \frac{\text{total correct} - \text{correct by predicting from base rate}}{\text{maximum correct} - \text{correct by predicting from base rate}} \tag{17.14}$$

17.6.8.14 Sensitivity (SN)

Sensitivity (SN) is also called true positive rate (TPR), hit rate (HR), or recall. Sensitivity is the conditional probability of a positive test given that the person has the condition: $P(R|C)$. Higher values reflect greater accuracy. The formula for calculating sensitivity is in Equation 17.15. As described in

Section Section 17.6.6.1, as the cutoff increases (becomes more conservative), sensitivity decreases. As the cutoff decreases, sensitivity increases.

$$\begin{aligned}\text{sensitivity (SN)} &= P(R|C) \\ &= \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{\text{TP}}{N \times \text{BR}} = 1 - \text{FNR}\end{aligned} \tag{17.15}$$

17.6.8.15 Specificity (SP)

Specificity (SP) is also called true negative rate (TNR) or selectivity. Specificity is the conditional probability of a negative test given that the person does not have the condition: $P(\text{not } R|\text{not } C)$. Higher values reflect greater accuracy. The formula for calculating specificity is in Equation 17.16. As described in Section Section 17.6.6.1, as the cutoff increases (becomes more conservative), specificity increases. As the cutoff decreases, specificity decreases.

$$\begin{aligned}\text{specificity (SP)} &= P(\text{not } R|\text{not } C) \\ &= \frac{\text{TN}}{\text{TN}+\text{FP}} = \frac{\text{TN}}{N(1-\text{BR})} = 1 - \text{FPR}\end{aligned} \tag{17.16}$$

17.6.8.16 False Negative Rate (FNR)

The false negative rate (FNR) is also called the miss rate. The false negative rate is the conditional probability of a negative test given that the person has the condition: $P(\text{not } R|C)$. Lower values reflect greater accuracy. The formula for calculating false negative rate is in Equation 17.17.

$$\begin{aligned}\text{false negative rate (FNR)} &= P(\text{not } R|C) \\ &= \frac{\text{FN}}{\text{FN}+\text{TP}} = \frac{\text{FN}}{N \times \text{BR}} = 1 - \text{TPR}\end{aligned} \tag{17.17}$$

17.6.8.17 False Positive Rate (FPR)

The false positive rate (FPR) is also called the false alarm rate (FAR) or fall-out. The false positive rate is the conditional probability of a positive test given that the person does not have the condition: $P(R|\text{not } C)$. Lower values reflect greater accuracy. The formula for calculating false positive rate is in Equation 17.18:

$$\begin{aligned}\text{false positive rate (FPR)} &= P(R|\text{not } C) \\ &= \frac{\text{FP}}{\text{FP}+\text{TN}} = \frac{\text{FP}}{N(1-\text{BR})} = 1 - \text{TNR}\end{aligned} \tag{17.18}$$

17.6.8.18 Positive Predictive Value (PPV)

The positive predictive value (PPV) is also called the positive predictive power (PPP) or precision. Many people confuse sensitivity ($P(R|C)$) with its inverse conditional probability, PPV ($P(C|R)$). PPV is the conditional probability of having the condition given a positive test: $P(C|R)$. Higher values reflect greater accuracy. The formula for calculating positive predictive value is in Equation 17.19.

PPV can be low even when sensitivity is high because it depends not only on sensitivity, but also on specificity and the base rate. Because PPV depends on the base rate, PPV is not an intrinsic property of a measure. The same measure will have a different PPV in different contexts with different base rates (Treat & Viken, 2023). As described in Section Section 17.6.6.1, as the base rate increases, PPV increases. As the base rate decreases, PPV decreases. PPV also differs as a function of the cutoff. As described in Section Section 17.6.6.1, as the cutoff increases (becomes more conservative), PPV increases. As the cutoff decreases (becomes more liberal), PPV decreases.

$$\begin{aligned}\text{positive predictive value (PPV)} &= P(C|R)\\ &= \frac{\text{TP}}{\text{TP}+\text{FP}} = \frac{\text{TP}}{N \times \text{SR}}\\ &= \frac{\text{sensitivity} \times \text{BR}}{\text{sensitivity} \times \text{BR} + [(1-\text{specificity}) \times (1-\text{BR})]}\end{aligned} \tag{17.19}$$

17.6.8.19 Negative Predictive Value (NPV)

The negative predictive value (NPV) is also called the negative predictive power (NPP). Many people confuse specificity ($P(\text{not } R|\text{not } C)$) with its inverse conditional probability, NPV ($P(\text{not } C|\text{not } R)$). NPV is the conditional probability of not having the condition given a negative test: $P(\text{not } C|\text{not } R)$. Higher values reflect greater accuracy. The formula for calculating negative predictive value is in Equation 17.20.

NPV can be low even when specificity is high because it depends not only on specificity, but also on sensitivity and the base rate. Because NPV depends on the base rate, NPV is not an intrinsic property of a measure. The same measure will have a different NPV in different contexts with different base rates (Treat & Viken, 2023). As described in Section Section 17.6.6.1, as the base rate increases, NPV decreases. As the base rate decreases, NPV increases. NPV also differs as a function of the cutoff. As described in Section Section 17.6.6.1, as the cutoff increases (becomes more conservative), NPV decreases. As the cutoff decreases (becomes more liberal), NPV decreases.

$$\begin{aligned}\text{negative predictive value (NPV)} &= P(\text{not } C|\text{not } R)\\ &= \frac{\text{TN}}{\text{TN}+\text{FN}} = \frac{\text{TN}}{N(1\text{ - SR})}\\ &= \frac{\text{specificity} \times (1-\text{BR})}{\text{specificity} \times (1-\text{BR}) + [(1-\text{sensitivity}) \times \text{BR})]}\end{aligned} \tag{17.20}$$

17.6.8.20 False Discovery Rate (FDR)

Many people confuse the false positive rate ($P(R|\text{not } C)$) with its inverse conditional probability, the false discovery rate ($P(\text{not } C|R)$). The false discovery rate (FDR) is the conditional probability of not having the condition given a positive test: $P(\text{not } C|R)$. Lower values reflect greater accuracy. The formula for calculating false discovery rate is in Equation 17.21.

$$\begin{aligned}\text{false discovery rate (FDR)} &= P(\text{not } C|R)\\ &= \frac{\text{FP}}{\text{FP}+\text{TP}} = 1-\text{PPV}\end{aligned} \tag{17.21}$$

17.6.8.21 False Omission Rate (FOR)

Many people confuse the false negative rate ($P(\text{not } R|C)$) with its inverse conditional probability, the false omission rate ($P(C|\text{not } R)$). The false omission rate (FOR) is the conditional probability of having the condition given a negative test: $P(C|\text{not } R)$. Lower values reflect greater accuracy. The formula for calculating false omission rate is in Section 17.6.8.21.

$$\begin{aligned}\text{false omission rate (FOR)} &= P(C|\text{not } R)\\ &= \frac{\text{FN}}{\text{FN}+\text{TN}} = 1-\text{NPV}\end{aligned}$$

{#sec-falseOmissionRate}

17.6.8.22 Youden's J Statistic

Youden's J statistic is also called Youden's Index or informedness. Youden's J statistic is the sum of sensitivity and specificity (and subtracting one). Higher values reflect greater accuracy. The formula for calculating Youden's J statistic is in Equation 17.22.

$$\text{Youden's J statistic} = \text{sensitivity} + \text{specificity} - 1 \tag{17.22}$$

17.6.8.23 Balanced Accuracy

Balanced accuracy is the average of sensitivity and specificity. Higher values reflect greater accuracy. The formula for calculating balanced accuracy is in Equation 17.23.

$$\text{balanced accuracy} = \frac{\text{sensitivity} + \text{specificity}}{2} \tag{17.23}$$

17.6.8.24 F-Score

The F-score combines precision (positive predictive value) and recall (sensitivity), where β indicates how many times more important sensitivity is than the positive predictive value. If sensitivity and the positive predictive value are equally important, $\beta = 1$, and the F-score is called the F_1 score. Higher values reflect greater accuracy. The formula for calculating the F-score is in Equation 17.24.

$$\begin{aligned} F_\beta &= (1+\beta^2) \cdot \frac{\text{positive predictive value} \cdot \text{sensitivity}}{(\beta^2 \cdot \text{positive predictive value}) + \text{sensitivity}} \\ &= \frac{(1+\beta^2) \cdot \text{TP}}{(1+\beta^2) \cdot \text{TP} + \beta^2 \cdot \text{FN} + \text{FP}} \end{aligned} \tag{17.24}$$

The formula for calculating the F_1 score is in Equation 17.25.

$$\begin{aligned} F_1 &= \frac{2 \cdot \text{positive predictive value} \cdot \text{sensitivity}}{(\text{positive predictive value}) + \text{sensitivity}} \\ &= \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FN} + \text{FP}} \end{aligned} \tag{17.25}$$

17.6.8.25 Matthews Correlation Coefficient (MCC)

The Matthews correlation coefficient (MCC) is also called the phi coefficient. It is a correlation coefficient between predicted and observed values from a binary classification. Higher values reflect greater accuracy. The formula for calculating the MCC is in Equation 17.26.

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}} \tag{17.26}$$

17.6.8.26 Diagnostic Odds Ratio

The diagnostic odds ratio is the odds of a positive test among people with the condition relative to the odds of a positive test among people without the

condition. Higher values reflect greater accuracy. The formula for calculating the diagnostic odds ratio is in Equation 17.27. If the predictor is bad, the diagnostic odds ratio could be less than one, and values can go up from there. If the diagnostic odds ratio is greater than 2, we take the odds ratio seriously because we are twice as likely to predict accurately than inaccurately. However, the diagnostic odds ratio ignores/hides base rates. When interpreting the diagnostic odds ratio, it is important to keep in mind the practical significance, because otherwise it is not very meaningful. Consider a risk factor that has a diagnostic odds ratio of 3 for tuberculosis, i.e., it puts you at 3 times as likely to develop tuberculosis. The prevalence of tuberculosis is relatively low. Assuming the prevalence of tuberculosis is less than 1/10th of 1%, your risk of developing tuberculosis is still very low even if the risk factor (with a diagnostic odds ratio of 3) is present.

$$\begin{aligned}\text{diagnostic odds ratio} &= \frac{\text{TP} \times \text{TN}}{\text{FP} \times \text{FN}} \\ &= \frac{\text{sensitivity} \times \text{specificity}}{(1 - \text{sensitivity}) \times (1 - \text{specificity})} \\ &= \frac{\text{PPV} \times \text{NPV}}{(1 - \text{PPV}) \times (1 - \text{NPV})} \\ &= \frac{\text{LR+}}{\text{LR}-}\end{aligned} \tag{17.27}$$

17.6.8.27 Diagnostic Likelihood Ratio

The diagnostic likelihood ratio is described in Section 16.5.2.1. There are two types of diagnostic likelihood ratios: the positive likelihood ratio and the negative likelihood ratio.

17.6.8.27.1 Positive Likelihood Ratio (LR+)

The positive likelihood ratio (LR+) is described in Section 16.5.2.1.1. The formula for calculating the positive likelihood ratio is in Equation 16.14.

17.6.8.27.2 Negative Likelihood Ratio (LR−)

The negative likelihood ratio (LR−) is described in Section 16.5.2.1.2. The formula for calculating the negative likelihood ratio is in Equation 16.14.

17.6.8.28 Posttest Odds

As presented in Equation 16.13, the posttest (or posterior) odds are equal to the pretest odds multiplied by the likelihood ratio. The posttest odds and posttest probability can be useful to calculate when the pretest probability is different from the pretest probability (or prevalence) of the classification. For instance, you might use a different pretest probability if a test result is already known and you want to know the updated posttest probability after conducting a second test. The formula for calculating posttest odds is in Equation 17.28.

$$\text{posttest odds} = \text{pretest odds} \times \text{likelihood ratio} \tag{17.28}$$

For calculating the posttest odds of a true positive compared to a false positive, we use the positive likelihood ratio below. We would use the negative likelihood ratio if we wanted to calculate the posttest odds of a false negative compared to a true negative.

17.6.8.29 Posttest Probability

The posttest probability is the probability of having the characteristic given a test result. When the base rate is used as the pretest probability, the posttest probability given a positive test is equal to positive predictive value. To convert odds to a probability, divide the odds by one plus the odds, as is in Equation 17.29.

$$\text{posttest probability} = \frac{\text{posttest odds}}{1 + \text{posttest odds}} \tag{17.29}$$

17.6.8.30 Mean Difference Between Predicted and Observed Values

The mean difference between predicted values versus observed values at a given cutoff is an index of miscalibration of predictions at that cutoff. It is called "calibration-in-the-small" (as opposed to calibration-in-the-large, which spans all cutoffs). Values closer to zero reflect greater accuracy. Values above zero indicate that the predicted values are, on average, greater than the observed values. Values below zero indicate that the observed values are, on average, greater than the predicted values.

17.7 Threshold-Independent Accuracy Indices

This section describes threshold-independent indexes of accuracy. That is, each index of accuracy described in this section provides a single numerical index of accuracy that aggregates the accuracy across all possible cutoffs. The `petersenlab`[3] package (Petersen, 2024a) contains the `accuracyOverall()` function that computes many threshold-independent accuracy indices.

17.7.1 General Prediction Accuracy

There are many metrics of general prediction accuracy. When thinking about which metric(s) may be best for a given problem, it is important to consider the purpose of the assessment. The estimates of general prediction accuracy are separated below into scale-dependent and scale-independent accuracy estimates.

17.7.1.1 Scale-Dependent Accuracy Estimates

The estimates of prediction accuracy described in this section are scale-dependent. These accuracy estimates depend on the unit of measurement and therefore cannot be compared across measures with different scales or across data sets.

17.7.1.1.1 Mean Error

Here, "error" (e) is the difference between the predicted and observed value for a given individual (i). Mean error (ME; also known as bias) is the mean difference between the predicted and observed values across individuals (i), that is, the mean of the errors across individuals (e_i). Values closer to zero reflect greater accuracy. If mean error is above zero, it indicates that predicted values are, on average, greater than observed values (i.e., overestimating errors). If mean error is below zero, it indicates that predicted values are, on average, less than observed values (i.e., underestimating errors). If both overestimating and under-estimating errors are present, however, they can cancel each other out. As a result, even with a mean error of zero, there can still be considerable error present. Thus, although mean error can be helpful for

[3] https://github.com/DevPsyLab/petersenlab

examining whether predictions systematically under- or over-estimate the actual scores, other forms of accuracy are necessary to examine the *extent* of error. The formula for mean error is in Equation 17.30:

$$\begin{aligned} \text{mean error} &= \frac{\sum_{i=1}^{n}(\text{predicted}_i - \text{observed}_i)}{n} \\ &= \text{mean}(e_i) \end{aligned} \tag{17.30}$$

17.7.1.1.2 Mean Absolute Error (MAE)

Mean absolute error (MAE) is the mean of the absolute value of differences between the predicted and observed values across individuals, that is, the mean of the absolute value of errors. Smaller MAE values (closer to zero) reflect greater accuracy. MAE is preferred over root mean squared error (RMSE) when you want to give equal weight to all errors and when the outliers have considerable impact. The formula for MAE is in Equation 17.31:

$$\begin{aligned} \text{mean absolute error (MAE)} &= \frac{\sum_{i=1}^{n}|\text{predicted}_i - \text{observed}_i|}{n} \\ &= \text{mean}(|e_i|) \end{aligned} \tag{17.31}$$

17.7.1.1.3 Mean Squared Error (MSE)

Mean squared error (MSE) is the mean of the square of the differences between the predicted and observed values across individuals, that is, the mean of the squared value of errors. Smaller MSE values (closer to zero) reflect greater accuracy. MSE penalizes larger errors more heavily than smaller errors (unlike MAE). However, MSE is sensitive to outliers and can be impacted if the errors are skewed. The formula for MSE is in Equation 17.32:

$$\begin{aligned} \text{mean squared error (MSE)} &= \frac{\sum_{i=1}^{n}(\text{predicted}_i - \text{observed}_i)^2}{n} \\ &= \text{mean}(e_i^2) \end{aligned} \tag{17.32}$$

17.7.1.1.4 Root Mean Squared Error (RMSE)

Root mean squared error (RMSE) is the square root of the mean of the square of the differences between the predicted and observed values across individuals, that is, the root mean squared value of errors. Smaller RMSE values (closer to zero) reflect greater accuracy. RMSE penalizes larger errors more heavily than smaller errors (unlike MAE). However, RMSE is sensitive to outliers and can be impacted if the errors are skewed. The formula for RMSE is in Equation 17.33:

$$\begin{aligned}\text{root mean squared error (RMSE)} &= \sqrt{\frac{\sum_{i=1}^{n}(\text{predicted}_i - \text{observed}_i)^2}{n}} \\ &= \sqrt{\text{mean}(e_i^2)}\end{aligned} \tag{17.33}$$

17.7.1.2 Scale-Independent Accuracy Estimates

The estimates of prediction accuracy described in this section are intended to be scale-*independent* (unit-free) so the accuracy estimates can be compared across measures with different scales or across data sets (Hyndman & Athanasopoulos, 2021).

17.7.1.2.1 Mean Percentage Error (MPE)

Mean percentage error (MPE) values closer to zero reflect greater accuracy. The formula for percentage error is in Equation 17.34:

$$\text{percentage error } (p_i) = \frac{100\% \times (\text{observed}_i - \text{predicted}_i)}{\text{observed}_i} \tag{17.34}$$

We then take the mean of the percentage errors to get MPE. The formula for MPE is in Equation 17.35:

$$\begin{aligned}\text{mean percentage error (MPE)} &= \frac{100\%}{n}\sum_{i=1}^{n}\frac{\text{observed}_i - \text{predicted}_i}{\text{observed}_i} \\ &= \text{mean(percentage error)} \\ &= \text{mean}(p_i)\end{aligned} \tag{17.35}$$

Note: MPE is undefined when one or more of the observed values equals zero, due to division by zero. The `accuracyOverall()` function of the `petersenlab`[4]

[4] https://github.com/DevPsyLab/petersenlab

package (Petersen, 2024a) provides the option in the function to drop undefined values so you can still generate an estimate of accuracy despite undefined values.

17.7.1.2.2 Mean Absolute Percentage Error (MAPE)

Smaller mean absolute percentage error (MAPE) values (closer to zero) reflect greater accuracy. The formula for MAPE is in Equation 17.36. MAPE is asymmetric because it overweights underestimates and underweights overestimates. MAPE can be preferable to symmetric mean absolute percentage error (sMAPE) if there are no observed values of zero and if you want to emphasize the importance of underestimates (relative to overestimates).

$$\begin{aligned}\text{mean absolute percentage error (MAPE)} &= \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{\text{observed}_i - \text{predicted}_i}{\text{observed}_i}\right| \\ &= \text{mean}(|\text{percentage error}|) \\ &= \text{mean}(|p_i|)\end{aligned} \tag{17.36}$$

Note: MAPE is undefined when one or more of the observed values equals zero, due to division by zero. The `accuracyOverall()` function of the `petersenlab`[5] package (Petersen, 2024a) provides the option in the function to drop undefined values so you can still generate an estimate of accuracy despite undefined values.

17.7.1.2.3 Symmetric Mean Absolute Percentage Error (sMAPE)

Unlike MAPE, symmetric mean absolute percentage error (sMAPE) is symmetric because it equally weights underestimates and overestimates. Smaller sMAPE values (closer to zero) reflect greater accuracy. The formula for sMAPE is in Equation 17.37:

$$\text{symmetric mean absolute percentage error (sMAPE)} = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|\text{predicted}_i - \text{observed}_i|}{|\text{predicted}_i| + |\text{observed}_i|} \tag{17.37}$$

Note: sMAPE is undefined when one or more of the individuals has a prediction–observed combination such that the sum of the absolute value of the predicted value and the absolute value of the observed value equals zero ($|\text{predicted}_i| + |\text{observed}_i|$), due to division by zero. The `accuracyOverall()`

[5] https://github.com/DevPsyLab/petersenlab

function of the `petersenlab`[6] package (Petersen, 2024a) provides the option in the function to drop undefined values so you can still generate an estimate of accuracy despite undefined values.

17.7.1.2.4 Mean Absolute Scaled Error (MASE)

Mean absolute scaled error (MASE) is described by (Hyndman & Athanasopoulos, 2021). Values closer to zero reflect greater accuracy.

The adapted formula for MASE with non-time series data is described here (https://stats.stackexchange.com/a/108963/20338)[7] (archived at https://perma.cc/G469-8NAJ). Scaled errors are calculated using Equation 17.38:

$$\begin{aligned}\text{scaled error}(q_i) &= \frac{\text{observed}_i - \text{predicted}_i}{\text{scaling factor}} \\ &= \frac{\text{observed}_i - \text{predicted}_i}{\frac{1}{n}\sum\limits_{i=1}^{n}|\text{observed}_i - \overline{\text{observed}}|}\end{aligned} \tag{17.38}$$

Then, we calculate the mean of the absolute value of the scaled errors to get MASE, as in Equation 17.39:

$$\begin{aligned}\text{mean absolute scaled error (MASE)} &= \frac{1}{n}\sum_{i=1}^{n}|q_i| \\ &= \text{mean}(|\text{scaled error}|) \\ &= \text{mean}(|q_i|)\end{aligned} \tag{17.39}$$

Note: MASE is undefined when the scaling factor is zero, due to division by zero. With non-time series data, the scaling factor is the average of the absolute value of individuals' observed scores minus the average observed score ($\frac{1}{n}\sum\limits_{i=1}^{n}|\text{observed}_i - \overline{\text{observed}}|$).

17.7.1.2.5 Root Mean Squared Log Error (RMSLE)

The squared log of the accuracy ratio is described by Tofallis (2015). The accuracy ratio is in Equation 17.40:

[6] https://github.com/DevPsyLab/petersenlab
[7] https://stats.stackexchange.com/a/108963/20338

$$\text{accuracy ratio} = \frac{\text{predicted}_i}{\text{observed}_i} \tag{17.40}$$

However, the accuracy ratio is undefined with observed or predicted values of zero, so it is common to modify it by adding 1 to the predictor and denominator, as in Equation 17.41:

$$\text{accuracy ratio} = \frac{\text{predicted}_i + 1}{\text{observed}_i + 1} \tag{17.41}$$

Squaring the log values keeps the values positive, such that smaller values (values closer to zero) reflect greater accuracy. Then we take the mean of the squared log values, which keeps the values positive, and calculate the square root of the mean squared log values to put them back on the (pre-squared) log metric. This is known as the root mean squared log error (RMSLE). Division inside the log is equal to subtraction outside the log. So, the formula can be reformulated with the subtraction of two logs, as in Equation 17.42:

$$\begin{aligned}\text{root mean squared log error (RMSLE)} &= \sqrt{\sum_{i=1}^{n} log\left(\frac{\text{predicted}_i + 1}{\text{observed}_i + 1}\right)^2} \\ &= \sqrt{\text{mean}\left[log\left(\frac{\text{predicted}_i + 1}{\text{observed}_i + 1}\right)^2\right]} \\ &= \sqrt{\text{mean}[log(\text{accuracy ratio})^2]} = \sqrt{\text{mean}\left\{[log(\text{predicted}_i + 1) - log(\text{actual}_i + 1)]^2\right\}}\end{aligned} \tag{17.42}$$

RMSLE can be preferable when the scores have a wide range of values and are skewed. RMSLE can help to reduce the impact of outliers. RMSLE gives more weight to smaller errors in the prediction of small observed values, while also penalizing larger errors in the prediction of larger observed values. It overweights underestimates and underweights overestimates.

There are other variations of prediction accuracy metrics that use the log of the accuracy ratio. One variation makes it similar to median symmetric percentage error (Morley et al., 2018).

Note: Root mean squared log error is undefined when one or more predicted values or actual values equals -1. When predicted or actual values are -1, this leads to $log(0)$, which is undefined. The `accuracyOverall()` function of the `petersenlab`[8] package (Petersen, 2024a) provides the option in the function to drop undefined values so you can still generate an estimate of accuracy despite undefined values.

17.7.1.2.6 Coefficient of Determination (R^2)

[8] https://github.com/DevPsyLab/petersenlab

The coefficient of determination (R^2) reflects the proportion of variance in the outcome (dependent) variable that is explained by the model predictions: $R^2 = \frac{\text{variance explained in } Y}{\text{total variance in } Y}$. Larger values indicate greater accuracy.

R^2 is commonly estimated in multiple regression, in which multiple predictors are allowed to predict one outcome.

17.7.1.2.6.1 Adjusted R^2 (R^2_{adj})

Adjusted R^2 is similar to the coefficient of determination, but it accounts for the number of predictors included in the regression model to penalize overfitting. Adjusted R^2 reflects the proportion of variance in the outcome (dependent) variable that is explained by the model predictions over and above what would be expected to be accounted for by chance, given the number of predictors in the model. Larger values indicate greater accuracy. The formula for adjusted R^2 is in Equation 11.4. Adjusted R^2 is described further in Section 11.5.

17.7.1.2.6.2 Predictive R^2

Predictive R^2 is described here: https://tomhopper.me/2014/05/16/can-we-do-better-than-r-squared/ (archived at https://perma.cc/BK8J-HFUK). Predictive R^2 penalizes overfitting, unlike traditional R^2. Larger values indicate greater accuracy.

17.7.2 Discrimination

When dealing with a categorical outcome, discrimination is the ability to separate events from non-events. When dealing with a continuous outcome, discrimination is the strength of the association between the predictor and the outcome. Threshold-dependent aspects of discrimination at a particular cutoff (e.g., sensitivity, specificity) are described in Section 17.6.

17.7.2.1 Area under the ROC curve (AUC)

The area under the ROC curve (AUC) is a general index of discrimination accuracy for a categorical outcome. It is also called the concordance (c) statistic. Larger values reflect greater discrimination accuracy. AUC was estimated using the `pROC` package (Robin et al., 2023).

17.7.2.2 Effect Size (β) of Regression

The effect size of a predictor, i.e., the standardized regression coefficient is called a beta (β) coefficient, is a general index of discrimination accuracy for a continuous outcome. Larger values reflect greater accuracy. We can obtain standardized regression coefficients by standardizing the predictors and outcome using the `scale()` function in `R`.

17.7.3 Calibration

When dealing with a categorical outcome, calibration is the degree to which a probabilistic estimate of an event reflects the true underlying probability of the event. When dealing with a continuous outcome, calibration is the degree to which the predicted values are close in value to the outcome values. The importance of examining calibration (in addition to discrimination) is described by Lindhiem et al. (2020). Calibration can be examined in several ways, including Spiegelhalter's z (see Section 17.7.3.2), and the mean difference between predicted and observed values at different binned thresholds as depicted graphically with a calibration plot (see Figure 17.7).

17.7.3.1 Calibration Plot

Calibration plots can be helpful for identifying miscalibration. A calibration plot depicts the predicted probability of an event on the x-axis, and the actual (observed) probability of the event on the y-axis. The predictions are binned into a certain number of groups (commonly 10). The diagonal line reflects predictions that are perfectly calibrated. To the extent that predictions deviate from the diagonal line, the predictions are miscalibrated.

Well-calibrated predictions are depicted in Figure 17.6:

```
# Specify data
examplePredictionsWellCalibrated <- seq(from = 0, to = 1, by = .1)
exampleOutcomesWellCalibrated <- seq(from = 0, to = 1, by = .1)

# Plot
plot(
  examplePredictionsWellCalibrated,
  exampleOutcomesWellCalibrated,
  xlim = c(0,1),
  ylim = c(0,1),
  xlab = "Predicted Probability",
  ylab = "Observed Proportion",
  bty = "l",
```

```
  type = "n")

lines(
  c(0,1),
  c(0,1),
  lwd = 2,
  col = "#377eb8")

points(
  examplePredictionsWellCalibrated,
  exampleOutcomesWellCalibrated,
  cex = 1.5,
  col = "#e41a1c",
  lwd = 2,
  type = "p")
```

Figure 17.6 Predictions that are Well-Calibrated. That is, the predicted values are close to the observed values.

The various types of general miscalibration are depicted in Figure 17.7:

```
# Specify data
examplePredictions <- seq(from = 0, to = 1, by = .1)
exampleOutcomes <- c(0, .15, .3, .4, .45, .5, .55, .6, .7, .85, 1)

overPrediction <- c(0, .02, .05, .1, .15, .2, .3, .4, .5, .7, 1)
underPrediction <- c(0, .3, .5, .6, .7, .8, .85, .9, .95, .98, 1)
overExtremity <- c(0, .3, .38, .42, .47, .5, .53, .58, .62, .7, 1)
underExtremity <- c(0, .05, .08, .11, .2, .5, .8, .89, .92, .95, 1)

# Plot
par(
```

```
  mfrow = c(2,2),
  mar = c(5,4,1,1) + 0.1) #margins: bottom, left, top, right

plot(
  examplePredictions,
  overExtremity,
  xlim = c(0,1),
  ylim = c(0,1),
  main = "Overextremity",
  xlab = "Predicted Probability",
  ylab = "Observed Proportion",
  bty = "l",
  cex = 1.5,
  col = "#e41a1c",
  type = "o")

lines(
  c(0,1),
  c(0,1),
  lwd = 2,
  col = "#377eb8")

plot(
  examplePredictions,
  underExtremity,
  xlim = c(0,1),
  ylim = c(0,1),
  main = "Underextremity",
  xlab = "Predicted Probability",
  ylab = "Observed Proportion",
  bty = "l",
  cex = 1.5,
  col = "#e41a1c",
  type = "o")

lines(
  c(0,1),
  c(0,1),
  lwd = 2,
  col = "#377eb8")

plot(
  examplePredictions,
  overPrediction,
```

```
  xlim = c(0,1),
  ylim = c(0,1),
  main = "Overprediction",
  xlab = "Predicted Probability",
  ylab = "Observed Proportion",
  bty = "l",
  cex = 1.5,
  col = "#e41a1c",
  type = "o")

lines(
  c(0,1),
  c(0,1),
  lwd = 2,
  col = "#377eb8")

plot(
  examplePredictions,
  underPrediction,
  xlim = c(0,1),
  ylim = c(0,1),
  main = "Underprediction",
  xlab = "Predicted Probability",
  ylab = "Observed Proportion",
  bty = "l",
  cex = 1.5,
  col = "#e41a1c",
  type = "o")

lines(
  c(0,1),
  c(0,1),
  lwd = 2,
  col = "#377eb8")
```

Figure 17.7 Types of Miscalibration. From Petersen (2024b) and Petersen (2024c).

However, predictions could also be miscalibrated in more specific ways. For instance, predictions could be well-calibrated at all predicted probabilities except for a given predicted probability (e.g., 20%). Or, the predictions could be miscalibrated but not systematically over- or underpredicted. Thus, it is important to evaluate a calibration plot to evaluate the extent to which the predictions are miscalibrated and the pattern of that miscalibration.

17.7.3.2 Spiegelhalter's z

Spiegelhalter's z was calculated using the `rms` package (Harrell, Jr., 2024). Smaller z values (and larger associated p-values) reflect greater calibration accuracy. A statistically significant Spiegelhalter's z ($p < .05$) indicates a

significant degree of miscalibration.

17.7.3.3 Calibration for predicting a continuous outcome

When predicting a continuous outcome, calibration of the predicted values in relation to the outcome values can be examined in multiple ways including:

- in a calibration plot, the extent to which the intercept is near zero and the slope is near one
- in a calibration plot, the extent to which the 95% confidence interval of the observed value, across all values of the predicted values, includes the diagonal reference line with an intercept of zero and a slope of one
- mean error
- mean absolute error
- mean squared error
- root mean squared error

With a plot of the predictions on the x-axis, and the outcomes on the y-axis (i.e., a calibration plot), calibration can be examined graphically as the extent to which the best-fit regression line has an intercept (alpha) close to zero and a slope (beta) close to one (Stevens & Poppe, 2020; Steyerberg & Vergouwe, 2014). The intercept is also called "calibration-in-the-large", whereas "calibration-in-the-small" refers to the extent to which the predicted values match the observed values at a specific predicted value (e.g., when the weather forecaster says that there is a 10% chance of rain, does it actually rain 10% of the time?). For predictions to be well calibrated, the intercept should be close to zero and the slope should be close to one. If the slope is close to one but the intercept is not close to zero (or the intercept is close to zero but the slope is not close to one), the predictions would not be considered well calibrated. The 95% confidence interval of the observed value, across all values of the predicted values, should include the diagonal reference line whose intercept is zero and whose slope is one.

For instance, based on the intercept and slope of the calibration plot in Figure INSERT, the predictions are not well calibrated, despite having a slope near one, because the 95% confidence interval of the intercept does not include zero. The best-fit line is the yellow line. The intercept from the best-fit line is positive, as shown in the regression equation. This is a case of underprediction, where the predicted values are consistently less than the observed values. The confidence interval of the observed value (i.e., the purple band) is the interval within which we have 95% confidence that the true observed value would lie for a given predicted value, based on the model The 95% prediction interval of the observed value (i.e., the dashed red lines) is the interval within which we would expect that 95% of future observations would lie for a given predicted value. The black diagonal line indicates the reference line with an intercept of

zero and a slope of one. The predictions would be significantly miscalibrated at a given level of the predicted values if the 95% confidence interval of the observed value does not include the reference line at that level of the predicted value. In this case, the 95% confidence interval of the observed value does not include the reference line (i.e., the actual observed value) at lower levels of the predicted values, so the predictions are miscalibrated lower levels of the predicted values.

Gold-standard recommendations include examining the predicted values in relation to the observed values using locally estimated scatterplot smoothing (LOESS) (Austin & Steyerberg, 2014), such as in Figure INSERT. We can examine whether the LOESS-based 95% confidence interval of the observed value at every level of the predicted values includes the diagonal reference line (i.e., the actual observed value). In this case, the 95% confidence interval of the observed value does not include the reference line at lower levels of the predicted values, so the predictions are miscalibrated at lower levels of the predicted values.

17.8 Integrating the Accuracy Indices

After computing the accuracy indices of discrimination and (2) calibration, it is then the task to integrate the indices to determine (a) which are the most accurate predictions for the given goals, and (b) whether additional improvements and refinements to the predictions need to be made. Each of the accuracy indices is computed differently and thus reward (and penalize) predictive (in)accuracy differently. Sometimes, the the accuracy indices will paint a consistent picture regarding which predictions are the most accurate. Other times, the accuracy indices may disagree about which predictions are most accurate.

In fantasy football, when evaluating the accuracy of seasonal projections, we care most about accurately distinguishing between higher levels of points (e.g., 200 vs 150) as opposed to lower levels of points (e.g., 0 vs 10). Thus, it can be helpful to punish larger errors more heavily than smaller errors, as RMSE (unlike MAE).

Thus, we would emphasize the following metrics:

- discrimination:
  - adjusted R^2
- calibration:

  - calibration plot

- general accuracy:
  - RMSE

If you focus on only one accuracy index, RMSE would be a good choice. However, I would also examine a calibration plot to evaluate whether predictions are poorly calibrated at higher levels of points. I would also examine ME—not to compare the accuracy of various predictions per se—but to determine whether predictions are systematically under- or overestimating actual points. If so, predictions may be able to be refined by adding or subtracting a constant to the predictions; however, this could worsen other accuracy indices, so it is important to conduct an iterative process of modifying then evaluating, then further modifiying and evaluating, etc. It may also be valuable to evaluate the accuracy of various subsets of the predictions. For instance, you might examine the predictive accuracy of players whose project points are greater than 100, to evaluate the accuracy of predictions specifically to distinguish between players at higher levels of points.

If we are making predictions about a categorical variable, we would emphasize the following metrics:

- discrimination:
  - area under the receiver operating curve
  - and, secondarily—depending on the particulary cutoff and the relative costs of false positives versus false negatives:
    * sensitivity
    * specificicity
    * positive predictive value
    * negative predictive value

- calibration:
  - calibration plot
  - Spiegelhalter's z
  - Mean difference between observed and predicted values

17.9 Theory Versus Empiricism

One question that inevitably arises when making predictions is the extent to which one should leverage theory versus empiricism. Theory involves conceptual claims of understanding how the causal system works (i.e., what influences

what). For example, use of theory in prediction might involve specification of the causal system that influences player performance, measurement of those factors, and the integration of that information to make a prediction. Empiricism involves "letting the data speak for themselves" and is an atheoretical approach. For example, empiricism might involve examining how thousands of variables are associated with the criterion of interest (e.g., fantasy points) and developing the best-fitting model based on those thousands of predictor variables.

Although the atheoretical approach can perform reasonably well, it can be improved by making better use of theory. An empirical result (e.g., a correlation) might not necessarily have a lot of meaning associated with it. As the maxim goes, correlation does not imply causation. Moreover, empiricism can lead to overfitting. So, empiricism is often not enough.

As Silver (2012) notes, "The numbers have no way of speaking for themselves. We speak for them. We imbue them with meaning." (p. 9). If we *understand* the variables in the system and how they influence each other, we can predict things more accurately than predicting for the sake of predicting. For instance, we have made great strides in the last decades when it comes to more accurate weather forecasts[9] (archived at https://perma.cc/PF8P-BT3D), including extreme weather events like hurricanes. These great strides have more to do with a better causal understanding of the weather system and the ability to conduct simulations of the atmosphere than merely because of big data (Silver, 2012). By contrast, other events are still incredibly difficult to predict, including earthquakes, in large part because we do not have a strong understanding of the system (and because we do not have ways of precisely measuring those causes because they occur at a depth below which we are realistically able to drill) (Silver, 2012).

At the same time, in the social and behavioral sciences, our theories of the causal processes that influence outcomes are not yet very strong. Indeed, I have misgivings calling them theories because they do not meet the traditional scientific standard for a theory. A scientific theory is an explanation of the natural world that is testable and falsifiable, and that has withstood rigorous scientific testing and scrutiny. In psychology (and other areas of social and behavioral sciences), our "theories" are more like conceptual frameworks. And these conceptual frameworks are often vague, do not make specific predictions of effects *and* noneffects, and do not hold up consistently when rigorously tested. As described by Meehl (1978):

> I consider it unnecessary to persuade you that most so-called

[9] https://www.npr.org/sections/money/2023/07/11/1186458991/should-we-invest-more-in-weather-forecasting-it-may-save-your-life

> "theories" in the soft areas of psychology (clinical, counseling, social, personality, community, and school psychology) are scientifically unimpressive and technologically worthless … Perhaps the easiest way to convince yourself is by scanning the literature of soft psychology over the last 30 years and noticing what happens to theories. Most of them suffer the fate that General MacArthur ascribed to old generals—They never die, they just slowly fade away. In the developed sciences, theories tend either to become widely accepted and built into the larger edifice of well-tested human knowledge or else they suffer destruction in the face of recalcitrant facts and are abandoned, perhaps regretfully as a "nice try." But in fields like personology and social psychology, this seems not to happen. There is a period of enthusiasm about a new theory, a period of attempted application to several fact domains, a period of disillusionment as the negative data come in, a growing bafflement about inconsistent and unreplicable empirical results, multiple resort to ad hoc excuses, and then finally people just sort of lose interest in the thing and pursue other endeavors. (pp. 806–807).

Even if we had strong theoretical understanding of the causal system that influences behavior, we would likely still have difficulty making accurate predictions because the field has largely relied on relatively crude instruments. According to one philosophical perspective known as LaPlace's demon, if we were able to know the exact conditions of everything in the universe, we would be able to know how the conditions would be in the future. This is an example of scientific determinism, where if you know the initial conditions, you also know the future. Other perspectives, such as quantum mechanics and chaos theory, would say that, even if we knew the initial conditions with 100% certainty, there would still be uncertainty in our understanding of the future. But assume, for a moment, that LaPlace's demon is true. A challenge in the social and behavioral sciences is that we have a relatively poor understanding of the initial conditions of the universe. Thus, our predictions would necessarily be probabilistic, similar to weather forecasts. Despite having a strong understanding of how weather systems behave, we have imperfect understanding of the initial conditions (e.g., the position and movement of all molecules) (Silver, 2012).

Theories tend to make grand conceptual claims that one observed variable influences another observed variable through a complex chain of intervening processes that are unobservable. Empiricism provides rich lower-level information, but lacks the broader picture. So, it seems, that we need both theory *and* empiricism. Theory and empiricism can—and should—inform each other.

17.10 Test Bias

Test bias refers to systematic error (in measurement, prediction, etc.) as a function of group membership that leads the same score to have different meaning for different groups. For instance, if the Wonderlic Contemporary Cognitive Ability Test is a strong predictor of performance for Quarterbacks but not for Running Backs, the test is biased. Test bias, including how to identify and address it, is described in Petersen (2024c)[10].

17.11 Ways to Improve Prediction Accuracy

On the whole, experts' predictions are inaccurate. Experts' predictions from many different domains tend to be inaccurate, including political scientists (Tetlock, 2017), physicians (Koehler et al., 2002), clinical psychologists (Oskamp, 1965), stock market traders and corporate financial officers (Skala, 2008), seismologists' predictions of earthquakes (Hough, 2016), economists' predictions about the economy (Makridakis et al., 2009), lawyers (Koehler et al., 2002), and business managers (Russo & Schoemaker, 1992). The most common pattern of experts' predictions is that they show overextremity, that is, their predictions have probability judgments that tend to be too extreme, as described in Section Section 17.3.2. Overextremity of experts' predictions reflects the overprecision type of overconfidence bias. The degree of confidence of a person's predictions is often not a good indicator of the accuracy of their predictions [and confidence and prediction accuracy are sometimes inversely associated; Silver (2012)]. Heuristics such as the anchoring and adjustment heuristic, cognitive biases such as confirmation bias (Hoch, 1985; Koriat et al., 1980), fallacies such as the base rate fallacy (Eddy, 1982; Koehler et al., 2002) could contribute to overconfidence of predictions. Poorly calibrated predictions are especially likely when the base rate is very low (e.g., suicide) or when the base rate is very high (Koehler et al., 2002).

Nevertheless, there are some domains that have shown greater predictive accuracy, from which we may learn what practices may lead to greater accuracy. For instance, experts have shown stronger predictive accuracy in weather forecasting (Murphy & Winkler, 1984), horse race betting (Johnson & Bruce, 2001), and playing the card game of bridge (Keren, 1987), but see Koehler et al. (2002) for exceptions.

[10] https://isaactpetersen.github.io/Principles-Psychological-Assessment/bias.html

Here are some potential ways to improve the accuracy (and honesty) of predictions and judgments:

- Provide appropriate anchoring of your predictions to the base rate of the phenomenon you are predicting. To the extent that the base rate of the event you are predicting is low, more extreme evidence should be necessary to consistently and accurately predict that the event will occur. Applying actuarial formulas and Bayes' theorem can help you appropriately weigh the base rate and evidence.
- Include multiple predictors, ideally from different measures and measurement methods. Include the predictors with the strongest validity based on theory of the causal process and based on criterion-related validity.
- When possible, aggregate multiple perspectives of predictions, especially predictions made independently (from different people/methods/etc.). The "wisdom of the crowd" is often more accurate than individuals' predictions, including predictions by so-called "experts" (Silver, 2012).
- A goal of prediction is to capture as much signal as possible and as little noise (error) as possible (Silver, 2012). Parsimony (i.e., not having too many predictors) can help reduce the amount of error variance captured by the prediction model. However, to accurately model complex systems like human behavior, complex models may be necessary. However, strong theory of the causal processes and dynamics may be necessary to develop accurate complex models.
- Although incorporating theory can be helpful, provide more weight to empiricism than to theory, until our theories and measures are stronger. Ideally, we would use theory to design a model that mirrors the causal system, with accurate measures of each process in the system, so we could make accurate predictions. However, as described in Section 17.9, our psychological theories of the causal processes that influence behavior are not yet very strong. Until we have stronger theories that specify the causal process for a given outcome, and until we have accurate measures of those causal processes, actuarial approaches are likely to be most accurate, as discussed in Chapter 15. At the same time, keep in mind that measures involving human behavior, and their resulting data, are often noisy. As a result, theoretically (conceptually) informed empirical approaches may lead to more accuracy than empiricism alone.
- Use an empirically validated and cross-validated statistical algorithm to combine information from the predictors in a formalized way. Give each predictor appropriate weight in the statistical algorithm, according to its strength of association with the outcome. Use measures with strong reliability and validity for assessing these processes to be used in the algorithm. Cross-validation will help reduce the likelihood that your model is fitting to noise and will maximize the likelihood that the model predicts accurately when applied to new data (i.e., the model's predictions accurately generalize), as described in Section 15.8.

- When presenting your predictions, acknowledge what you do not know.
- Express your predictions in terms of probabilistic estimates and present the uncertainty in your predictions with confidence intervals [even though bolder, more extreme predictions tend to receive stronger television ratings; Silver (2012)].
- Qualify your predictions by identifying and noting counter-examples that would not be well fit by your prediction model, such as extreme cases, edge cases, and "broken leg" (Meehl, 1957) cases.
- Provide clear, consistent, and timely feedback on the outcomes of the predictions to the people making the predictions (Bolger & Önkal-Atay, 2004).
- Be self-critical about your predictions. Update your judgments based on their accuracy, rather than trying to confirm your beliefs (Atanasov et al., 2020).
- In addition to considering the accuracy of the prediction, consider the quality of the prediction *process*, especially when random chance is involved to a degree, such as in poker and fantasy football (Silver, 2012).
- Work to identify and mitigate potential blindspots; be aware of cognitive biases and fallacies, such as confirmation bias and the base rate fallacy.
- Evaluate for the possibility of test bias. Correct for any test bias.

17.12 Conclusion

When the base rate of a behavior is very low or very high, you can be highly accurate in predicting the behavior by predicting from the base rate. Thus, you cannot judge how accurate your prediction is until you know how accurate your predictions would be by random chance. Moreover, maximizing percent accuracy may not be the ultimate goal because different errors have different costs. Though there are many indices of accuracy, there are two general types of accuracy: discrimination and calibration. Discrimination accuracy is frequently evaluated with the area under the receiver operating characteristic curve, or with sensitivity and specificity, or with standardized regression coefficients or the coefficient of determination. Calibration accuracy is frequently evaluated graphically and with various indices. Sensitivity and specificity depend on the cutoff. It is important to evaluate both discrimination and calibration when evaluating prediction accuracy.

18

Mythbusters: Putting Fantasy Football Beliefs/Anecdotes to the Test

18.1 Getting Started

18.1.1 Load Packages

```
library("petersenlab")
library("nflreadr")
library("lme4")
library("lmerTest")
library("MuMIn")
library("emmeans")
library("tidyverse")
```

18.1.2 Specify Package Options

```
emm_options(lmerTest.limit = 100000)
emm_options(pbkrtest.limit = 100000)
```

18.1.3 Load Data

```
load(file = "./data/nfl_playerContracts.RData")
load(file = "./data/player_stats_weekly.RData")
load(file = "./data/player_stats_seasonal.RData")
load(file = "./data/nfl_espnQBR_seasonal.RData")
load(file = "./data/nfl_espnQBR_weekly.RData")
```

We created the `player_stats_weekly.RData` and `player_stats_seasonal.RData` objects in Section 4.4.3.

18.2 Do Players Perform Better in their Contract Year?

Considerable speculation exists regarding whether players perform better in their last year of their contract (i.e., their "contract year"). Fantasy football talking heads and commentators frequently discuss the benefit of selecting players who are in their contract year, because it supposedly means that player has more motivation to perform well so they get a new contract and get paid more. To our knowledge, no peer-reviewed studies have examined this question for football players. One study found that National Basketball Association (NBA) players improved in field goal percentage, points, and player efficiency rating (but not other statistics: rebounds, assists, steals, or blocks) from their pre-contract year to their contract year, and that Major League Baseball (MLB) players improved in runs batted in (RBIs; but not other statistics: batting average, slugging percentage, on base percentage, home runs, fielding percentage) from their pre-contract year to their contract year (White & Sheldon, 2014). Other casual analyses have been examined contract-year performance of National Football League (NFL) players, including articles in 2012[1] (archived here[2]) and 2022[3] (archived here[4]).

Let's examine the question empirically. In order to do that, we have to make some assumptions/constraints. In this example, we will make the following constraints:

- We will determine a player's contract year programmatically based on the year the contract was signed. For instance, if a player signed a 3-year contract in 2015, their contract would expire in 2018, and thus their contract year would be 2017. Note: this is a coarse way of determining a player's contract year because it could depend on when during the year the player's contract is signed. If we were submitting this analysis as a paper to a scientific journal, it would be important to verify each player's contract year.
- We will examine performance in all seasons since 2011, beginning when most data for player contracts are available.

[1] https://www.4for4.com/2012/preseason/2012-contract-year-players-and-myth-increased-production
[2] https://perma.cc/CT3F-QN5E
[3] https://www.4for4.com/2022/preseason/do-players-perform-better-fantasy-football-contract-year
[4] https://perma.cc/F4F5-7RQZ

- For maximum statistical power to detect an effect if a contract year effect exists, we will examine all seasons for a player (since 2011), not just their contract year and their pre-contract year.
- To ensure a more fair, apples-to-apples comparison of the games in which players played, we will examine *per-game* performance (except for yards per carry, which is based on $\frac{\text{rushing yards}}{\text{carries}}$ from the entire season).
- We will examine regular season games only (no postseason).
- To ensure we do not make generalization about a player's performance in a season from a small sample, the player has to play at least 5 games in a given season for that player–season combination to be included in analysis.

For analysis, the same player contributes multiple observations of performance (i.e., multiple seasons) due to the longitudinal nature of the data. Inclusion of multiple data points from the same player would violate the assumption of multiple regression that all observations are independent. Thus, we use mixed-effects models that allow nonindependent observations. In our mixed-effects models, we include a random intercept for each player, to allow our model to account for players' differing level of performance. We examine two mixed-effects models for each outcome variable: one model that accounts for the effects of age and experience, and one model that does not.

The model that does not account for the effects of age and experience includes:

a) random intercepts to allow the model to estimate a different starting point for each player
b) a fixed effect for whether the player is in a contract year

The model that accounts for the effects of age and experience includes:

a) random intercepts to allow the model to estimate a different starting point for each player
b) random linear slopes (i.e., random effect of linear age) to allow the model to estimate a different form of change for each player
c) a fixed quadratic effect of age to allow for curvilinear effects
d) a fixed effect of experience
e) a fixed effect for whether the player is in a contract year

```
# Subset to remove players without a year signed
nfl_playerContracts_subset <- nfl_playerContracts %>%
  dplyr::filter(!is.na(year_signed) & year_signed != 0)

# Determine the contract year for a given contract
nfl_playerContracts_subset$contractYear <- nfl_playerContracts_subset$year_signed + nfl_playerCont

# Arrange contracts by player and year_signed
```

```
nfl_playerContracts_subset <- nfl_playerContracts_subset %>%
  dplyr::group_by(player, position) %>%
  dplyr::arrange(player, position, -year_signed) %>%
  dplyr::ungroup()

# Determine if the player played in the original contract year
nfl_playerContracts_subset <- nfl_playerContracts_subset %>%
  dplyr::group_by(player, position) %>%
  dplyr::mutate(
    next_contract_start = lag(year_signed)) %>%
  dplyr::ungroup() %>%
  dplyr::mutate(
    played_in_contract_year = ifelse(
      is.na(next_contract_start) | contractYear < next_contract_start,
      TRUE,
      FALSE))

# Check individual players
#nfl_playerContracts_subset %>%
#  dplyr::filter(player == "Aaron Rodgers") %>%
#  dplyr::select(player:years, contractYear, next_contract_start, played_in_contract_year)
#
#nfl_playerContracts_subset %>%
#  dplyr::filter(player %in% c("Jared Allen", "Aaron Rodgers")) %>%
#  dplyr::select(player:years, contractYear, next_contract_start, played_in_contract_year)

# Subset data
nfl_playerContractYears <- nfl_playerContracts_subset %>%
  dplyr::filter(played_in_contract_year == TRUE) %>%
  dplyr::filter(position %in% c("QB","RB","WR","TE")) %>%
  dplyr::select(player, position, team, contractYear) %>%
  dplyr::mutate(merge_name = nflreadr::clean_player_names(player, lowercase = TRUE)) %>%
  dplyr::rename(season = contractYear) %>%
  dplyr::mutate(contractYear = 1)

# Merge with weekly and seasonal stats data
player_stats_weekly_offense <- player_stats_weekly_offense %>%
  dplyr::mutate(merge_name = nflreadr::clean_player_names(player_display_name, lowercase = TRUE))
#nfl_actualStats_offense_seasonal <- nfl_actualStats_offense_seasonal %>%
#  mutate(merge_name = nflreadr::clean_player_names(player_display_name, lowercase = TRUE))

player_statsContracts_offense_weekly <- dplyr::full_join(
  player_stats_weekly_offense,
  nfl_playerContractYears,
```

```
  by = c("merge_name", "position_group" = "position", "season")
) %>%
  dplyr::filter(position_group %in% c("QB","RB","WR","TE"))

#player_statsContracts_offense_seasonal <- full_join(
#  player_stats_seasonal_offense,
#  nfl_playerContractYears,
#  by = c("merge_name", "position_group" = "position", "season")
#) %>%
#  filter(position_group %in% c("QB","RB","WR","TE"))

player_statsContracts_offense_weekly$contractYear[which(is.na(player_statsContracts_offense_weekly
#player_statsContracts_offense_seasonal$contractYear[which(is.na(player_statsContracts_offense_sea

#player_statsContracts_offense_weekly$contractYear <- factor(
#  player_statsContracts_offense_weekly$contractYear,
#  levels = c(0, 1),
#  labels = c("no", "yes"))

#player_statsContracts_offense_seasonal$contractYear <- factor(
#  player_statsContracts_offense_seasonal$contractYear,
#  levels = c(0, 1),
#  labels = c("no", "yes"))

player_statsContracts_offense_weekly <- player_statsContracts_offense_weekly %>%
  dplyr::arrange(merge_name, season, season_type, week)

#player_statsContracts_offense_seasonal <- player_statsContracts_offense_seasonal %>%
#  arrange(merge_name, season)

player_statsContractsSubset_offense_weekly <- player_statsContracts_offense_weekly %>%
  dplyr::filter(season_type == "REG")

#table(nfl_playerContracts$year_signed) # most contract data is available beginning in 2011

# Calculate Per Game Totals
player_statsContracts_seasonal <- player_statsContractsSubset_offense_weekly %>%
  dplyr::group_by(player_id, season) %>%
  dplyr::summarise(
    player_display_name = petersenlab::Mode(player_display_name),
    position_group = petersenlab::Mode(position_group),
    age = min(age, na.rm = TRUE),
    years_of_experience = min(years_of_experience, na.rm = TRUE),
    rushing_yards = sum(rushing_yards, na.rm = TRUE), # season total
```

```
    carries = sum(carries, na.rm = TRUE), # season total
    rushing_epa = mean(rushing_epa, na.rm = TRUE),
    receiving_yards = mean(receiving_yards, na.rm = TRUE),
    receiving_epa = mean(receiving_epa, na.rm = TRUE),
    contractYear = mean(contractYear, na.rm = TRUE),
    games = n(),
    .groups = "drop_last"
  ) %>%
  dplyr::mutate(
    player_id = as.factor(player_id),
    ypc = rushing_yards / carries,
    contractYear = factor(
      contractYear,
      levels = c(0, 1),
      labels = c("no", "yes")
    ))

player_statsContracts_seasonal[sapply(player_statsContracts_seasonal, is.infinite)] <- NA

player_statsContracts_seasonal$ageCentered20 <- player_statsContracts_seasonal$age - 20
player_statsContracts_seasonal$ageCentered20Quadratic <- player_statsContracts_seasonal$ageCentere

# Merge with seasonal fantasy points data
```

18.2.1 QB

First, we prepare the data by merging and performing additional processing:

```
# Merge with QBR data
nfl_espnQBR_weekly$merge_name <- paste(nfl_espnQBR_weekly$name_first, nfl_espnQBR_weekly$name_last
  nflreadr::clean_player_names(., lowercase = TRUE)

nfl_contractYearQBR_weekly <- nfl_playerContractYears %>%
  dplyr::filter(position == "QB") %>%
  dplyr::full_join(
    .,
    nfl_espnQBR_weekly,
    by = c("merge_name","team","season")
  )

nfl_contractYearQBR_weekly$contractYear[which(is.na(nfl_contractYearQBR_weekly$contractYear))] <- 
#nfl_contractYearQBR_weekly$contractYear <- factor(
#  nfl_contractYearQBR_weekly$contractYear,
```

```
#  levels = c(0, 1),
#  labels = c("no", "yes"))

nfl_contractYearQBR_weekly <- nfl_contractYearQBR_weekly %>%
  dplyr::arrange(merge_name, season, season_type, game_week)

nfl_contractYearQBRsubset_weekly <- nfl_contractYearQBR_weekly %>%
  dplyr::filter(season_type == "Regular") %>%
  dplyr::arrange(merge_name, season, season_type, game_week) %>%
  mutate(
    player = coalesce(player, name_display),
    position = "QB") %>%
  group_by(merge_name, player_id) %>%
  fill(player, .direction = "downup")

# Merge with age and experience
nfl_contractYearQBRsubset_weekly <- player_statsContractsSubset_offense_weekly %>%
  dplyr::filter(position == "QB") %>%
  dplyr::select(merge_name, season, week, age, years_of_experience) %>%
  full_join(
    nfl_contractYearQBRsubset_weekly,
    by = c("merge_name","season", c("week" = "game_week"))
  ) %>% select(player_id, season, week, player, everything()) %>%
  arrange(player_id, season, week)

#hist(nfl_contractYearQBRsubset_weekly$qb_plays) # players have at least 20 dropbacks per game

# Calculate Per Game Totals
nfl_contractYearQBR_seasonal <- nfl_contractYearQBRsubset_weekly %>%
  dplyr::group_by(merge_name, season) %>%
  dplyr::summarise(
    age = min(age, na.rm = TRUE),
    years_of_experience = min(years_of_experience, na.rm = TRUE),
    qbr = mean(qbr_total, na.rm = TRUE),
    pts_added = mean(pts_added, na.rm = TRUE),
    epa_pass = mean(pass, na.rm = TRUE),
    qb_plays = sum(qb_plays, na.rm = TRUE), # season total
    contractYear = mean(contractYear, na.rm = TRUE),
    games = n(),
    .groups = "drop_last"
  ) %>%
  dplyr::mutate(
    contractYear = factor(
      contractYear,
```

```
      levels = c(0, 1),
      labels = c("no", "yes")
    ))

nfl_contractYearQBR_seasonal[sapply(nfl_contractYearQBR_seasonal, is.infinite)] <- NA

nfl_contractYearQBR_seasonal$ageCentered20 <- nfl_contractYearQBR_seasonal$age - 20
nfl_contractYearQBR_seasonal$ageCentered20Quadratic <- nfl_contractYearQBR_seasonal$ageCentered20

nfl_contractYearQBR_seasonal <- nfl_contractYearQBR_seasonal %>%
  group_by(merge_name) %>%
  mutate(player_id = as.factor(as.character(cur_group_id())))

nfl_contractYearQBRsubset_seasonal <- nfl_contractYearQBR_seasonal %>%
  dplyr::filter(
    games >= 5, # keep only player-season combinations in which QBs played at least 5 games
    season >= 2011) # keep only seasons since 2011 (when most contract data are available)
```

Then, we analyze the data. Below is a mixed model that examines whether a player has a higher QBR per game when they are in a contract year compared to when they are not in a contract year.

```
mixedModel_qbr <- lmerTest::lmer(
  qbr ~ contractYear + (1 | player_id),
  data = nfl_contractYearQBR_seasonal,
  control = lmerControl(optimizer = "bobyqa")
)

summary(mixedModel_qbr)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: qbr ~ contractYear + (1 | player_id)
   Data: nfl_contractYearQBR_seasonal
Control: lmerControl(optimizer = "bobyqa")

REML criterion at convergence: 8905.7

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.2554 -0.5475  0.0907  0.5738  3.2547

Random effects:
 Groups    Name        Variance Std.Dev.
```

```
 player_id (Intercept) 111.6    10.56
 Residual              198.6    14.09
Number of obs: 1063, groups:  player_id, 253

Fixed effects:
                 Estimate Std. Error        df t value Pr(>|t|)
(Intercept)      44.26920    0.87356 231.38798  50.677   <2e-16 ***
contractYearyes   0.05517    1.19528 943.21436   0.046    0.963
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr)
contrctYrys -0.240
```

```r
MuMIn::r.squaredGLMM(mixedModel_qbr)
```

```
              R2m       R2c
[1,] 1.506369e-06 0.3597179
```

```r
emmeans::emmeans(mixedModel_qbr, "contractYear")
```

```
 contractYear emmean    SE  df lower.CL upper.CL
 no             44.3 0.874 262     42.5     46.0
 yes            44.3 1.302 752     41.8     46.9

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
mixedModelAge_qbr <- lmerTest::lmer(
  qbr ~ contractYear + ageCentered20 + ageCentered20Quadratic + years_of_experience + (1 + ageCent
  data = nfl_contractYearQBR_seasonal,
  control = lmerControl(optimizer = "bobyqa")
)

summary(mixedModelAge_qbr)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: qbr ~ contractYear + ageCentered20 + ageCentered20Quadratic +
    years_of_experience + (1 + ageCentered20 | player_id)
   Data: nfl_contractYearQBR_seasonal
Control: lmerControl(optimizer = "bobyqa")
```

```
REML criterion at convergence: 6844.3

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.6110 -0.5094  0.0710  0.5457  3.4258

Random effects:
 Groups    Name          Variance Std.Dev. Corr
 player_id (Intercept)   122.2762 11.0579
           ageCentered20   0.6852  0.8277  -0.38
 Residual                166.4307 12.9008
Number of obs: 831, groups:  player_id, 167

Fixed effects:
                        Estimate Std. Error        df t value Pr(>|t|)
(Intercept)           39.48193    2.37393 163.40575  16.631  < 2e-16 ***
contractYearyes          0.27777    1.22839 730.82229   0.226 0.821169
ageCentered20           2.05900    0.72486 267.29207   2.841 0.004850 **
ageCentered20Quadratic -0.08794   0.02550  96.09781  -3.449 0.000837 ***
years_of_experience     -0.55190    0.64905 241.75297  -0.850 0.395989
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) cntrcY agCn20 agC20Q
contrctYrys  0.046
ageCentrd20 -0.764 -0.089
agCntrd20Qd  0.701  0.035 -0.535
yrs_f_xprnc  0.227 -0.001 -0.713 -0.157
```

```r
MuMIn::r.squaredGLMM(mixedModelAge_qbr)
```

```
            R2m       R2c
[1,] 0.02113517 0.4411265
```

```r
emmeans::emmeans(mixedModelAge_qbr, "contractYear")
```

```
 contractYear emmean   SE  df lower.CL upper.CL
 no             45.2 1.12 153     43.0     47.4
 yes            45.5 1.37 359     42.8     48.2

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
mixedModel_ptsAdded <- lmerTest::lmer(
  pts_added ~ contractYear + (1 | player_id),
  data = nfl_contractYearQBR_seasonal,
  control = lmerControl(optimizer = "bobyqa")
)

summary(mixedModel_ptsAdded)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: pts_added ~ contractYear + (1 | player_id)
   Data: nfl_contractYearQBR_seasonal
Control: lmerControl(optimizer = "bobyqa")

REML criterion at convergence: 4855.7

Scaled residuals:
    Min      1Q  Median      3Q     Max
-4.7037 -0.4894  0.0854  0.5362  4.4263

Random effects:
 Groups    Name        Variance Std.Dev.
 player_id (Intercept) 2.571    1.604
 Residual              4.331    2.081
Number of obs: 1063, groups:  player_id, 253

Fixed effects:
                 Estimate Std. Error        df t value Pr(>|t|)
(Intercept)      -0.85134    0.13127 219.79804  -6.485 5.76e-10 ***
contractYearyes  -0.08925    0.17680 931.73837  -0.505    0.614
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr)
contrctYrys -0.236
```

```r
MuMIn::r.squaredGLMM(mixedModel_ptsAdded)
```

```
              R2m       R2c
[1,] 0.0001770825 0.3726302
```

```r
emmeans::emmeans(mixedModel_ptsAdded, "contractYear")
```

```
 contractYear emmean    SE  df lower.CL upper.CL
 no           -0.851 0.131 262    -1.11   -0.593
 yes          -0.941 0.194 745    -1.32   -0.560

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
mixedModelAge_ptsAdded <- lmerTest::lmer(
  pts_added ~ contractYear + ageCentered20 + ageCentered20Quadratic + years_of_experience + (1 + a
  data = nfl_contractYearQBR_seasonal,
  control = lmerControl(optimizer = "bobyqa")
)

summary(mixedModelAge_ptsAdded)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: pts_added ~ contractYear + ageCentered20 + ageCentered20Quadratic +
    years_of_experience + (1 + ageCentered20 | player_id)
   Data: nfl_contractYearQBR_seasonal
Control: lmerControl(optimizer = "bobyqa")

REML criterion at convergence: 3768.4

Scaled residuals:
    Min      1Q  Median      3Q     Max
-4.9543 -0.4823  0.0788  0.5074  4.4513

Random effects:
 Groups    Name          Variance Std.Dev. Corr
 player_id (Intercept)   3.42877  1.8517
           ageCentered20 0.01121  0.1059   -0.49
 Residual                4.04121  2.0103
Number of obs: 831, groups:  player_id, 167

Fixed effects:
                       Estimate Std. Error         df t value Pr(>|t|)
(Intercept)         -1.538867   0.373292 147.195811  -4.122 6.24e-05 ***
contractYearyes        -0.101309   0.190415 725.357555  -0.532  0.59486
ageCentered20          0.230585   0.112569 251.840267   2.048  0.04156 *
ageCentered20Quadratic -0.012099  0.003865  67.516183  -3.130  0.00258 **
years_of_experience     0.006180   0.100457 220.010278   0.062  0.95101
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) cntrcY agCn20 agC20Q
contrctYrys  0.046
ageCentrd20 -0.762 -0.090
agCntrd20Qd  0.698  0.046 -0.537
yrs_f_xprnc  0.239 -0.003 -0.721 -0.151
```

```
MuMIn::r.squaredGLMM(mixedModelAge_ptsAdded)
```

```
            R2m       R2c
[1,] 0.01239464 0.4186167
```

```
emmeans::emmeans(mixedModelAge_ptsAdded, "contractYear")
```

```
 contractYear emmean    SE  df lower.CL upper.CL
 no           -0.657 0.168 158   -0.988   -0.325
 yes          -0.758 0.210 383   -1.171   -0.345

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```
mixedModel_epaPass <- lmerTest::lmer(
  epa_pass ~ contractYear + (1 | player_id),
  data = nfl_contractYearQBR_seasonal,
  control = lmerControl(optimizer = "bobyqa")
)

summary(mixedModel_epaPass)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: epa_pass ~ contractYear + (1 | player_id)
   Data: nfl_contractYearQBR_seasonal
Control: lmerControl(optimizer = "bobyqa")

REML criterion at convergence: 4534.4

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.0166 -0.5102  0.0389  0.5643  4.3629
```

```
Random effects:
 Groups    Name        Variance Std.Dev.
 player_id (Intercept) 2.449    1.565
 Residual              3.053    1.747
Number of obs: 1063, groups:  player_id, 253

Fixed effects:
                Estimate Std. Error       df t value Pr(>|t|)
(Intercept)       1.0788     0.1217 239.5143   8.862  < 2e-16 ***
contractYearyes   0.3957     0.1497 921.4232   2.644  0.00833 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr)
contrctYrys -0.213
```

```r
MuMIn::r.squaredGLMM(mixedModel_epaPass)
```

```
             R2m       R2c
[1,] 0.004348227 0.4474718
```

```r
emmeans::emmeans(mixedModel_epaPass, "contractYear")
```

```
 contractYear emmean    SE  df lower.CL upper.CL
 no             1.08 0.122 262    0.839     1.32
 yes            1.47 0.172 699    1.137     1.81

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
mixedModelAge_epaPass <- lmerTest::lmer(
  epa_pass ~ contractYear + ageCentered20 + ageCentered20Quadratic + years_of_experience + (1 | pl
  data = nfl_contractYearQBR_seasonal,
  control = lmerControl(optimizer = "bobyqa")
)

summary(mixedModelAge_epaPass)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: epa_pass ~ contractYear + ageCentered20 + ageCentered20Quadratic +
```

```
    years_of_experience + (1 | player_id)
   Data: nfl_contractYearQBR_seasonal
Control: lmerControl(optimizer = "bobyqa")

REML criterion at convergence: 3452.2

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.4462 -0.5233  0.0693  0.5448  4.4517

Random effects:
 Groups    Name        Variance Std.Dev.
 player_id (Intercept) 1.922    1.386
 Residual              2.805    1.675
Number of obs: 831, groups:  player_id, 167

Fixed effects:
                       Estimate Std. Error        df t value Pr(>|t|)
(Intercept)           0.485918   0.291166 674.959069   1.669  0.09561 .
contractYearyes        0.053588   0.157236 752.652519   0.341  0.73334
ageCentered20         0.237823   0.089958 489.474418   2.644  0.00846 **
ageCentered20Quadratic  -0.004309   0.002830 777.320235  -1.522  0.12835
years_of_experience    -0.052657   0.082951 292.647453  -0.635  0.52606
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) cntrcY agCn20 agC20Q
contrctYrys  0.048
ageCentrd20 -0.747 -0.091
agCntrd20Qd  0.685  0.054 -0.492
yrs_f_xprnc  0.264 -0.006 -0.751 -0.169
```

```r
MuMIn::r.squaredGLMM(mixedModelAge_epaPass)
```

```
            R2m       R2c
[1,] 0.04725467 0.4347018
```

```r
emmeans::emmeans(mixedModelAge_epaPass, "contractYear")
```

```
 contractYear emmean    SE  df lower.CL upper.CL
 no             1.75 0.139 183     1.47     2.02
 yes            1.80 0.174 411     1.46     2.14
```

```
Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```
# Placeholder for model predicting fantasy points
```

18.2.2 RB

```
player_statsContractsRB_seasonal <- player_statsContracts_seasonal %>%
  dplyr::filter(
    position_group == "RB",
    games >= 5, # keep only player-season combinations in which QBs played at least 5 games
    season >= 2011) # keep only seasons since 2011 (when most contract data are available)
```

```
mixedModel_ypc <- lmerTest::lmer(
  ypc ~ contractYear + (1 | player_id),
  data = player_statsContractsRB_seasonal,
  control = lmerControl(optimizer = "bobyqa")
)

summary(mixedModel_ypc)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: ypc ~ contractYear + (1 | player_id)
   Data: player_statsContractsRB_seasonal
Control: lmerControl(optimizer = "bobyqa")

REML criterion at convergence: 4821.1

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.8768 -0.4660  0.0118  0.4845  6.4861

Random effects:
 Groups    Name        Variance Std.Dev.
 player_id (Intercept) 0.3923   0.6263
 Residual              1.1417   1.0685
Number of obs: 1512, groups:  player_id, 482

Fixed effects:
                 Estimate Std. Error        df t value Pr(>|t|)
(Intercept)     3.911e+00  4.484e-02 4.835e+02  87.227   <2e-16 ***
```

```
contractYearyes 4.942e-02  6.971e-02 1.423e+03   0.709    0.478
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr)
contrctYrys -0.350
```

```r
MuMIn::r.squaredGLMM(mixedModel_ypc)
```

```
              R2m       R2c
[1,] 0.0003022763 0.2559658
```

```r
emmeans::emmeans(mixedModel_ypc, "contractYear")
```

```
 contractYear emmean     SE   df lower.CL upper.CL
 no             3.91 0.0449  557     3.82     4.00
 yes            3.96 0.0685 1146     3.83     4.09

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
mixedModelAge_ypc <- lmerTest::lmer(
  ypc ~ contractYear + ageCentered20 + ageCentered20Quadratic + years_of_experience + (1 + ageCent
  data = player_statsContractsRB_seasonal,
  control = lmerControl(optimizer = "bobyqa")
)

summary(mixedModelAge_ypc)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: ypc ~ contractYear + ageCentered20 + ageCentered20Quadratic +
    years_of_experience + (1 + ageCentered20 | player_id)
   Data: player_statsContractsRB_seasonal
Control: lmerControl(optimizer = "bobyqa")

REML criterion at convergence: 4080.3

Scaled residuals:
    Min      1Q  Median      3Q     Max
-4.0210 -0.4563 -0.0008  0.4722  6.4138
```

```
Random effects:
 Groups    Name          Variance Std.Dev. Corr
 player_id (Intercept)   1.105261 1.05131
           ageCentered20 0.009768 0.09884  -0.80
 Residual                0.995099 0.99755
Number of obs: 1296, groups:  player_id, 361

Fixed effects:
                        Estimate Std. Error         df t value Pr(>|t|)
(Intercept)          4.289e+00  1.623e-01  4.344e+02  26.422   <2e-16 ***
contractYearyes       1.801e-01  7.591e-02  1.106e+03   2.373   0.0178 *
ageCentered20        -5.189e-02  5.784e-02  4.129e+02  -0.897   0.3702
ageCentered20Quadratic -1.755e-03  3.710e-03  1.690e+02  -0.473   0.6368
years_of_experience   -3.617e-03  3.761e-02  2.744e+02  -0.096   0.9235
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) cntrcY agCn20 agC20Q
contrctYrys  0.165
ageCentrd20 -0.870 -0.195
agCntrd20Qd  0.757  0.131 -0.771
yrs_f_xprnc  0.179 -0.018 -0.492 -0.104
```

```r
MuMIn::r.squaredGLMM(mixedModelAge_ypc)
```

```
            R2m       R2c
[1,] 0.02629169 0.3708211
```

```r
emmeans::emmeans(mixedModelAge_ypc, "contractYear")
```

```
 contractYear emmean     SE  df lower.CL upper.CL
 no             3.91 0.0535 396     3.81     4.02
 yes            4.09 0.0740 899     3.95     4.24

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
mixedModel_epaRush <- lmerTest::lmer(
  rushing_epa ~ contractYear + (1 | player_id),
  data = player_statsContractsRB_seasonal,
  control = lmerControl(optimizer = "bobyqa")
)

summary(mixedModel_epaRush)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: rushing_epa ~ contractYear + (1 | player_id)
   Data: player_statsContractsRB_seasonal
Control: lmerControl(optimizer = "bobyqa")

REML criterion at convergence: 4286.4

Scaled residuals:
    Min      1Q  Median      3Q     Max
-4.6919 -0.5186  0.0887  0.6130  3.4722

Random effects:
 Groups    Name        Variance Std.Dev.
 player_id (Intercept) 0.1026   0.3204
 Residual              0.9051   0.9514
Number of obs: 1512, groups:  player_id, 482

Fixed effects:
                  Estimate Std. Error         df t value Pr(>|t|)
(Intercept)       -0.65667    0.03298  577.83232 -19.912   <2e-16 ***
contractYearyes    0.04577    0.05916 1508.85908   0.774    0.439
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr)
contrctYrys -0.430
```

```r
MuMIn::r.squaredGLMM(mixedModel_epaRush)
```

```
              R2m       R2c
[1,] 0.0003945932 0.1021977
```

```r
emmeans::emmeans(mixedModel_epaRush, "contractYear")
```

```
 contractYear emmean    SE   df lower.CL upper.CL
 no           -0.657 0.033  575   -0.722   -0.592
 yes          -0.611 0.054 1054   -0.717   -0.505

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
mixedModelAge_epaRush <- lmerTest::lmer(
  rushing_epa ~ contractYear + ageCentered20 + ageCentered20Quadratic + years_of_experience + (1 +
  data = player_statsContractsRB_seasonal,
  control = lmerControl(optimizer = "bobyqa")
)

summary(mixedModelAge_epaRush)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: rushing_epa ~ contractYear + ageCentered20 + ageCentered20Quadratic +
    years_of_experience + (1 + ageCentered20 | player_id)
   Data: player_statsContractsRB_seasonal
Control: lmerControl(optimizer = "bobyqa")

REML criterion at convergence: 3678.1

Scaled residuals:
    Min      1Q  Median      3Q     Max
-4.8023 -0.5071  0.0638  0.5993  3.1974

Random effects:
 Groups    Name          Variance Std.Dev. Corr
 player_id (Intercept)   0.235283 0.48506
           ageCentered20 0.002966 0.05446  -0.74
 Residual                0.872259 0.93395
Number of obs: 1296, groups:  player_id, 361

Fixed effects:
                        Estimate Std. Error         df t value Pr(>|t|)
(Intercept)        -6.805e-01  1.271e-01  2.878e+02  -5.352 1.78e-07 ***
contractYearyes     9.073e-02  6.661e-02  1.146e+03   1.362  0.17345
ageCentered20       7.440e-02  4.479e-02  2.832e+02   1.661  0.09781 .
ageCentered20Quadratic -2.896e-03  3.078e-03  1.201e+02  -0.941  0.34868
years_of_experience   -7.127e-02  2.631e-02  3.153e+02  -2.709  0.00712 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) cntrcY agCn20 agC20Q
contrctYrys  0.150
ageCentrd20 -0.888 -0.201
agCntrd20Qd  0.801  0.124 -0.820
yrs_f_xprnc  0.070 -0.013 -0.372 -0.148
```

```
MuMIn::r.squaredGLMM(mixedModelAge_epaRush)
```

```
            R2m       R2c
[1,] 0.01477437 0.1432941
```

```
emmeans::emmeans(mixedModelAge_epaRush, "contractYear")
```

```
 contractYear emmean     SE  df lower.CL upper.CL
 no           -0.656 0.0382 382   -0.732   -0.581
 yes          -0.566 0.0588 903   -0.681   -0.450

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```
# Placeholder for model predicting fantasy points
```

18.2.3 WR/TE

```
player_statsContractsWRTE_seasonal <- player_statsContracts_seasonal %>%
  dplyr::filter(
    position_group %in% c("WR","TE"),
    games >= 5, # keep only player-season combinations in which QBs played at least 5 games
    season >= 2011) # keep only seasons since 2011 (when most contract data are available)
```

```
mixedModel_receivingYards <- lmerTest::lmer(
  receiving_yards ~ contractYear + (1 | player_id),
  data = player_statsContractsWRTE_seasonal,
  control = lmerControl(optimizer = "bobyqa")
)

summary(mixedModel_receivingYards)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: receiving_yards ~ contractYear + (1 | player_id)
   Data: player_statsContractsWRTE_seasonal
Control: lmerControl(optimizer = "bobyqa")

REML criterion at convergence: 27094.2
```

```
Scaled residuals:
    Min      1Q  Median      3Q     Max 
-4.7762 -0.5668 -0.0870  0.5265  4.5000 

Random effects:
 Groups    Name        Variance Std.Dev.
 player_id (Intercept) 233.9    15.29   
 Residual              190.1    13.79   
Number of obs: 3180, groups:  player_id, 937

Fixed effects:
                 Estimate Std. Error        df t value Pr(>|t|)    
(Intercept)       29.2630     0.6020 1147.8658  48.613  < 2e-16 ***
contractYearyes   -3.7991     0.6301 2752.6783  -6.029 1.87e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr)
contrctYrys -0.254
```

```r
MuMIn::r.squaredGLMM(mixedModel_receivingYards)
```

```
             R2m       R2c
[1,] 0.006779332 0.5547323
```

```r
emmeans::emmeans(mixedModel_receivingYards, "contractYear")
```

```
 contractYear emmean    SE   df lower.CL upper.CL
 no             29.3 0.602 1034     28.1     30.4
 yes            25.5 0.753 1931     24.0     26.9

Degrees-of-freedom method: kenward-roger 
Confidence level used: 0.95
```

```r
mixedModelAge_receivingYards <- lmerTest::lmer(
  receiving_yards ~ contractYear + ageCentered20 + ageCentered20Quadratic + years_of_experience +
  data = player_statsContractsWRTE_seasonal,
  control = lmerControl(optimizer = "bobyqa")
)

summary(mixedModelAge_receivingYards)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
```

```
lmerModLmerTest]
Formula:
receiving_yards ~ contractYear + ageCentered20 + ageCentered20Quadratic +
    years_of_experience + (1 + ageCentered20 | player_id)
   Data: player_statsContractsWRTE_seasonal
Control: lmerControl(optimizer = "bobyqa")

REML criterion at convergence: 23687.2

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.8960 -0.5509 -0.0850  0.5057  3.8002

Random effects:
 Groups    Name          Variance Std.Dev. Corr
 player_id (Intercept)   499.008  22.338
           ageCentered20   6.314   2.513   -0.67
 Residual                142.636  11.943
Number of obs: 2815, groups:  player_id, 742

Fixed effects:
                        Estimate Std. Error         df t value Pr(>|t|)
(Intercept)           17.65659   1.76256 1077.55573  10.018  < 2e-16 ***
contractYearyes       -3.38701   0.63921 2230.82411  -5.299 1.28e-07 ***
ageCentered20          3.71523   0.62760 1567.44179   5.920 3.95e-09 ***
ageCentered20Quadratic  -0.47157   0.03218 1157.02734 -14.653  < 2e-16 ***
years_of_experience     2.37600   0.49010  945.79109   4.848 1.46e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) cntrcY agCn20 agC20Q
contrctYrys  0.107
ageCentrd20 -0.834 -0.149
agCntrd20Qd  0.656  0.056 -0.630
yrs_f_xprnc  0.335  0.042 -0.680 -0.063
```

```
MuMIn::r.squaredGLMM(mixedModelAge_receivingYards)
```

```
           R2m       R2c
[1,] 0.1092205 0.7317998
```

```
emmeans::emmeans(mixedModelAge_receivingYards, "contractYear")
```

```
 contractYear emmean    SE   df lower.CL upper.CL
```

```
 no             28.9 0.752  807     27.4     30.4
 yes            25.5 0.838 1219     23.9     27.2

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
mixedModel_epaReceiving <- lmerTest::lmer(
  receiving_epa ~ contractYear + (1 | player_id),
  data = player_statsContractsWRTE_seasonal,
  control = lmerControl(optimizer = "bobyqa")
)

summary(mixedModel_epaReceiving)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula: receiving_epa ~ contractYear + (1 | player_id)
   Data: player_statsContractsWRTE_seasonal
Control: lmerControl(optimizer = "bobyqa")

REML criterion at convergence: 10538.7

Scaled residuals:
    Min      1Q  Median      3Q     Max
-4.1703 -0.5759 -0.0615  0.5309  3.9493

Random effects:
 Groups    Name        Variance Std.Dev.
 player_id (Intercept) 0.5242   0.724
 Residual              1.2749   1.129
Number of obs: 3178, groups:  player_id, 937

Fixed effects:
                  Estimate Std. Error         df t value Pr(>|t|)
(Intercept)        0.71352    0.03535 1269.78569  20.185   <2e-16 ***
contractYearyes   -0.11717    0.04936 3028.50007  -2.374   0.0177 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr)
contrctYrys -0.355
```

```r
MuMIn::r.squaredGLMM(mixedModel_epaReceiving)
```

```
             R2m       R2c
[1,] 0.001528258 0.2924573
```

```r
emmeans::emmeans(mixedModel_epaReceiving, "contractYear")
```

```
 contractYear emmean     SE   df lower.CL upper.CL
 no            0.714 0.0354 1092    0.644    0.783
 yes           0.596 0.0495 2184    0.499    0.693

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
mixedModelAge_epaReceiving <- lmerTest::lmer(
  receiving_epa ~ contractYear + ageCentered20 + ageCentered20Quadratic + years_of_experience + (1
  data = player_statsContractsWRTE_seasonal,
  control = lmerControl(optimizer = "bobyqa")
)

summary(mixedModelAge_epaReceiving)
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method [
lmerModLmerTest]
Formula:
receiving_epa ~ contractYear + ageCentered20 + ageCentered20Quadratic +
    years_of_experience + (1 + ageCentered20 | player_id)
   Data: player_statsContractsWRTE_seasonal
Control: lmerControl(optimizer = "bobyqa")

REML criterion at convergence: 9344.9

Scaled residuals:
    Min      1Q  Median      3Q     Max
-4.2448 -0.5720 -0.0470  0.5351  3.7843

Random effects:
 Groups    Name          Variance Std.Dev. Corr
 player_id (Intercept)   0.903632 0.95060
           ageCentered20 0.007471 0.08644  -0.62
 Residual                1.224389 1.10652
Number of obs: 2814, groups:  player_id, 742
```

```
Fixed effects:
                        Estimate Std. Error         df t value Pr(>|t|)
(Intercept)           3.021e-01  1.154e-01  8.511e+02   2.619 0.008974 **
contractYearyes      -1.846e-01  5.458e-02  2.582e+03  -3.383 0.000728 ***
ageCentered20         1.699e-01  3.966e-02  9.818e+02   4.283 2.02e-05 ***
ageCentered20Quadratic -1.355e-02  2.245e-03  3.953e+02  -6.038 3.60e-09 ***
years_of_experience    2.165e-02  2.771e-02  8.205e+02   0.781 0.434945
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) cntrcY agCn20 agC20Q
contrctYrys  0.123
ageCentrd20 -0.857 -0.193
agCntrd20Qd  0.745  0.089 -0.722
yrs_f_xprnc  0.206  0.056 -0.550 -0.113
```

```r
MuMIn::r.squaredGLMM(mixedModelAge_epaReceiving)
```

```
            R2m      R2c
[1,] 0.01951911 0.351717
```

```r
emmeans::emmeans(mixedModelAge_epaReceiving, "contractYear")
```

```
 contractYear emmean     SE   df lower.CL upper.CL
 no            0.807 0.0415  791    0.726    0.889
 yes           0.623 0.0535 1732    0.518    0.728

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```r
# Placeholder for model predicting fantasy points
```

18.2.4 QB/RB/WR/TE

```r
# Placeholder for model predicting fantasy points
# Include player position as a covariate
```

18.3 Conclusion

19

Modern Portfolio Theory

19.1 Getting Started

19.1.1 Load Packages

```
library("quantmod")
library("fPortfolio")
library("tidyverse")
```

19.2 Overview

19.3 Fantasy Football is Like Stock Picking

Selecting players for your fantasy team is like picking stocks. In both fantasy football and the stock market, your goal is to pick assets (i.e., players/stocks) that will perform best and that others undervalue. But what is the best way to do that? Below, we discuss approaches to picking players/stocks.

19.3.1 The Wisdom of the Crowd (or Market)

In picking players, there are various approaches one could take. You could do lots of research to pick players/stocks with strong fundamentals that you think will do particularly well next year. By picking these players/stocks, you are predicting that they will *outperform* their expectations. However, all of your information is likely already reflected in the current valuation of the

player/stock, so your prediction is basically a gamble. This is evidenced by the fact that people do not reliably beat the crowd/market.

Even so-called experts do not beat the market reliably. There is little consistency in the performance of mutual fund managers over time. In the book, "The Drunkard's Walk: How Randomness Rules Our Lives", Mlodinow (2008) reported essentially no correlation between performance of the top mutual funds in a five-year period with their performance over the subsequent five years. That is, the best funds in a one period were not necessarily the best funds in another period. This suggests that mutual fund managers differ in great part because of luck or chance rather than reliable skill. In any given year, some mutual funds will do better than other mutual funds. But this overperformance in a given year likely reflects more randomness than skill. That is likely why a cat beat professional investors in a stock market challenge[1] (archived at https://perma.cc/R3XU-K6J8). Although our sample size is much smaller with fantasy football projections, there also appears to be little consistency in fantasy football sites' rank in accuracy over time [INSERT], suggesting that the projection sources are not reliably better than each other (or the crowd) over time.

The market reflects all of the knowledge of the crowd. One common misconception is that if you go with the market, you will receive "average" returns (by "average", I mean that you will be in the 50th percentile among investors). This is not true—it has been shown that most mutual funds (about 80%) underperform the average returns of the stock market. So, by going with the market average, you will likely perform better than the "average" fund/investor. Consistent with this, crowd-averaged fantasy football projections tend to be more accurate than any individual's projection: INSERT This evidence is consistent with the notion of the wisdom of the crowd, described in Section 22.3. Moreover, even if the stock market is relatively accurate ("efficient") in terms of valuing stocks based on all (publicly) available information (i.e., the efficient market hypothesis), your fantasy football league is likely not. Thus, it may be effective to use crowd-based projections to identify players who are undervalued by your league.

19.3.2 Diversification

Modern portfolio theory (mean-variance theory) is a framework for determining the optimal composition of an investment portfolio to maximize expected returns for a given level of risk. Here, *risk* refers to the variability (e.g., standard deviation or variance) of returns across time. Given two portfolios with the same expected returns over time, people will prefer the "safer" portfolio—that is, the portfolio with less variability/volatility across time. One of the

[1] http://www.npr.org/blogs/money/2013/01/14/169326326/housecat-beats-investors-in-stock-market-challenge

powerful notions of modern portfolio theory is that, through diversification, one can achieve lower risk with the same expected returns. In investing, *diversification* involves owning multiple asset classes (e.g., domestic and international stocks and bonds), with the goal of having asset classes that are either uncorrelated or negatively correlated. That is, owning different types of assets is safer than owning only one type. If you have too much money in one asset and that asset tanks, you will lose your money. In other words, you do not want to put all of your eggs in one basket. By owning different asset classes, you can limit your downside risk without sacrificing much in terms of expected return.

This lesson can also apply to fantasy football. When assembling a team, you are essentially putting together a portfolio of assets (i.e., team of players). As with stocks, each player has an expected return (i.e., projection) and a degree of risk. In fantasy football, a player's risk might be quantified in terms of the variability of projected scores for a player across projection sources (e.g., Projection Source A, Source B, Souce C, etc.), or as historical game-to-game variability. Variability of projected scores for a player across projection sources could reflect the *uncertainty* of projections for a player. Variability of historical (actual) fantasy points across games could reflect many factors, including risks due to injuries, situational changes (e.g., being traded to a new team or changes in team composition such as due to the acquisition of new players on the team), game scripts, and the tendency for the player to be "boom-or-bust" (e.g., if they are highly dependent on scoring touchdowns or long receptions for fantasy points). All things equal, we want to minimize our risk for a given level of expected returns. That way, we have the best chance of winning any given week. For the same level of expected returns, higher risk teams might have a few amazing games, but their teams might fall flat in other weeks. That is, for a given (high) rate of return, you are best off in the long run (i.e., over the course of a season) with a lower risk team[2] compared to a higher risk team (archived at https://perma.cc/NE35-G6LR).

In terms of diversification, it can be helpful to diversify in multiple ways. First, it can be helpful not to rely on just one or two "stud" players. If they are on bye or have a down week, your team is more likely to suffer. Also, there are risks in picking multiple offensive players from the same team. If you draft your starting Quarterback and Wide Receiver from the same team (e.g., the Cowboys), you are exposing your fantasy team to considerable risk. For instance, if you have the Quarterback and Wide Receiver from the same team, and the team has a poor offensive outing, that will have a greater impact. You can limit your downside risk by diversifying—drafting players from different teams. That way if the Cowboys' offense does poorly in a given week, your fantasy team will not be as affected. Having multiple players on a juggernaut offense can be a boon, but it can be challenging to predict which offense will

[2] https://eng.wealthfront.com/2012/01/17/moneyball-using-modern-portfolio-theory-to-win-your-fantasy-sports-league/

lead the league.

However, sometimes having two players on the same team might be beneficial because some positions may be uncorrelated or even negatively correlated, which can also reduce risk. For instance, the performance of the Tight End and Running Back on the same team tends to be slightly negatively correlated, so it might not be a bad idea to start the Tight End and Running Back from the same team. For a correlation matrix of all positions on the team, see: https://assets-global.website-files.com/5f1af76ed86d6771ad48324b/607a4434a565aa7763bd1312_AndyAsh-Sharpstack-RPpaper.pdf (archived at https://perma.cc/JQ6G-KSRT).

Another important idea from modern portfolio theory is that, if you want to achieve higher returns, you may be able to by accepting additional—and the right combination of—risk. In general, risk is positively correlated with return. That is, receiving higher returns generally requires taking on additional risk—at least as long as we stay along the efficient frontier, described next.

19.4 The Efficient Frontier of a Stock Portfolio

The ultimate goal in fantasy football is to draft players for your starting lineup that provide the most projected points (i.e., the highest returns) and the smallest downside risk. That is, your goal is to achieve the optimal portfolio at a given level of risk, depending on how much risk you are willing to tolerate. One of the key tools in modern portfolio theory for identifying the optimal portfolio (for a given risk level) is the efficient frontier. The efficient frontier is a visual depiction of the maximum expected returns for a given level of risk (where risk is the variability in returns over time). The efficient frontier is helpful for identifying the optimal portfolio—the optimal combination and weighting of assets—for a given risk level. Anything below the efficient frontier is considered inefficient (i.e., lower-than-maximum returns for a given level of risk).

In the example below, we use historical returns (since YEAR) as the expected future returns. However, using historical returns as the expected future returns is risky because, as described in the common disclaimer, "Past performance does not guarantee future results." If you select a relatively short period of historical returns, you may be selecting a period when the stock performed particularly well. When evaluating historical returns it is preferable to evaluate long time horizons and to evaluate how the stock performed during period of both boom (i.e., "bull markets") and bust (i.e., "bear markets", such as in a recession).

19.4.1 Download Historical Stock Prices

```
symbols <- c(
  "AAPL",  # Apple
  "MSFT",  # Microsoft
  "GOOGL", # Google
  "AMZN",  # Amazon
  "META",  # Meta/Facebook
  "V",     # Visa
  "DIS",   # Disney
  "NKE",   # Nike
  "TSLA")  # Tesla

quantmod::getSymbols(symbols)
```

```
[1] "AAPL"  "MSFT"  "GOOGL" "AMZN"  "META"  "V"    "DIS"   "NKE"   "TSLA"
```

19.4.2 Calculate Stock Returns

```
prices <- do.call(
  merge,
  lapply(
    symbols,
    function(sym) quantmod::Cl(get(sym))))

returns <- na.omit(
  TTR::ROC(
    prices,
    type = "discrete"))

returns_ts <- timeSeries::as.timeSeries(returns)
```

19.4.3 Create Portfolio

```
portfolioSpec <- fPortfolio::portfolioSpec()

fPortfolio::setNFrontierPoints(portfolioSpec) <- 1000
```

19.4.4 Determine the Efficient Frontier

```
efficientFrontier <- fPortfolio::portfolioFrontier(
  returns_ts,
  spec = portfolioSpec)

efficientFrontier
```

```
Title:
 MV Portfolio Frontier
 Estimator:         covEstimator
 Solver:            solveRquadprog
 Optimize:          minRisk
 Constraints:       LongOnly
 Portfolio Points:  5 of 1000

Portfolio Weights:
   AAPL.Close MSFT.Close GOOGL.Close AMZN.Close META.Close V.Close DIS.Close
1      0.0000    0.0000     0.0000    0.0000    0.0000  0.0000   1.0000
250    0.1352    0.1291     0.1038    0.0518    0.0000  0.2998   0.1516
500    0.1029    0.2048     0.0000    0.1237    0.0791  0.2486   0.0000
750    0.0000    0.1188     0.0000    0.1443    0.1580  0.0000   0.0000
1000   0.0000    0.0000     0.0000    0.0000    0.0000  0.0000   0.0000
     NKE.Close TSLA.Close
1       0.0000     0.0000
250     0.1135     0.0152
500     0.0000     0.2409
750     0.0000     0.5789
1000    0.0000     1.0000

Covariance Risk Budgets:
   AAPL.Close MSFT.Close GOOGL.Close AMZN.Close META.Close V.Close DIS.Close
1      0.0000    0.0000     0.0000    0.0000    0.0000  0.0000   1.0000
250    0.1396    0.1340     0.1060    0.0549    0.0000  0.3020   0.1373
500    0.0765    0.1573     0.0000    0.1066    0.0723  0.1599   0.0000
750    0.0000    0.0430     0.0000    0.0662    0.0810  0.0000   0.0000
1000   0.0000    0.0000     0.0000    0.0000    0.0000  0.0000   0.0000
     NKE.Close TSLA.Close
1       0.0000     0.0000
250     0.1067     0.0195
500     0.0000     0.4274
750     0.0000     0.8098
1000    0.0000     1.0000
```

```
Target Returns and Risks:
       mean    Cov   CVaR    VaR
1    0.0004 0.0165 0.0377 0.0239
250  0.0008 0.0128 0.0303 0.0197
500  0.0013 0.0162 0.0373 0.0255
750  0.0017 0.0245 0.0547 0.0368
1000 0.0022 0.0356 0.0776 0.0509

Description:
 Fri Aug 30 22:30:14 2024 by user:
```

```r
# Extract the coordinates of individual assets
asset_means <- colMeans(returns)
asset_sd <- apply(returns, 2, sd)

# Add some padding to plot limits (so ticker symbols don't get cut off)
xlim <- range(asset_sd) * c(0.9, 1.1)
ylim <- range(asset_means) * c(0.9, 1.1)

xlim[1] <- 0
ylim[1] <- 0

# Set scientific notation penalty
options(scipen = 999)

plot(
  efficientFrontier,
  which = c(
    1,  # efficient frontier
    3,  # tangency portfolio
    4), # risk/return of individual assets
  control = list(
    xlim = xlim,
    ylim = ylim
  ))

# Add text labels for individual assets
points(
  asset_sd,
  asset_means,
  col = "red",
  pch = 19)

text(
```

```
  asset_sd,
  asset_means,
  labels = symbols,
  pos = 4,
  cex = 0.8,
  col = "black")
```

Figure 19.1 Efficient Frontier for a Stock Portfolio.

19.4.5 Identify the Optimal Weights

19.4.5.1 Tangency Portfolio

The tangency portfolio is the portfolio with the highest Sharpe ratio (i.e., the highest ratio of return to risk). In other words, it is the portfolio with the greatest risk-adjusted returns.

```
# Find the tangency portfolio (portfolio with the highest Sharpe ratio)
tangencyPortfolio <- fPortfolio::tangencyPortfolio(
  data = returns_ts,
  spec = portfolioSpec)

# Extract optimal weights
tangencyPortfolio_optimalWeights <- fPortfolio::getWeights(tangencyPortfolio)
tangencyPortfolio_optimalWeights
```

```
 AAPL.Close  MSFT.Close GOOGL.Close  AMZN.Close  META.Close     V.Close
0.113269063 0.202646410 0.008102524 0.118396042 0.070145970 0.276304490
  DIS.Close   NKE.Close  TSLA.Close
0.000000000 0.000000000 0.211135501
```

```r
# Output the results
summary(tangencyPortfolio)
```

```
Title:
 MV Tangency Portfolio
 Estimator:         covEstimator
 Solver:            solveRquadprog
 Optimize:          minRisk
 Constraints:       LongOnly

Portfolio Weights:
 AAPL.Close  MSFT.Close GOOGL.Close  AMZN.Close  META.Close     V.Close
     0.1133      0.2026      0.0081      0.1184      0.0701      0.2763
  DIS.Close   NKE.Close  TSLA.Close
     0.0000      0.0000      0.2111

Covariance Risk Budgets:
 AAPL.Close  MSFT.Close GOOGL.Close  AMZN.Close  META.Close     V.Close
     0.0896      0.1649      0.0061      0.1070      0.0668      0.1923
  DIS.Close   NKE.Close  TSLA.Close
     0.0000      0.0000      0.3733

Target Returns and Risks:
  mean    Cov   CVaR    VaR
0.0012 0.0157 0.0362 0.0246

Description:
 Fri Aug 30 22:30:14 2024 by user:
```

Weights

Weighted Returns

19.4.5.2 Portfolio with Max Return at a Given Risk Level

```
# Define target risk levels
targetRisks <- seq(0, 0.3, by = 0.01)

# Initialize storage for optimal portfolios
optimalPortfolios <- list()
optimalWeights_list <- list()

# Find optimal weightings for each target risk level
for (risk in targetRisks) {
  # Create a portfolio optimization specification with the target risk
  portfolioSpec <- fPortfolio::portfolioSpec()
  fPortfolio::setTargetRisk(portfolioSpec) <- risk

  # Solve for the maximum return at this target risk
  optimal_portfolio <- fPortfolio::maxreturnPortfolio(
    returns_ts,
    spec = portfolioSpec)

  # Store the optimal portfolio
  optimalPortfolios[[as.character(risk)]] <- optimal_portfolio

  # Store the optimal portfolio weights with risk level
  optimal_weights <- fPortfolio::getWeights(optimal_portfolio)
```

```
  optimalWeights_list[[as.character(risk)]] <- c(RiskLevel = risk, optimal_weights)
}

optimalWeightsByRisk <- dplyr::bind_rows(optimalWeights_list)
optimalWeightsByRisk
```

19.5 The Efficient Frontier of a Fantasy Team

In fantasy football, the efficient frontier can be helpful for identifying the optimal players to draft for a given risk level (and potentially within the salary cap). It can also be helpful for identifying potential trades. In this way, modern portfolio theory and the efficient frontier can be helpful for arbitrage—buying and selling the same asset (in this case, player) to take advantage of different prices for the same asset. That is, you could buy low and, for players who outperform expectations, sell high—in the form of a trade.

19.5.1 Based on Variability Across Projection Sources

19.5.2 Based on Historical Game-to-Game Variability

https://eng.wealthfront.com/2012/01/17/moneyball-using-modern-portfolio-theory-to-win-your-fantasy-sports-league/ (archived at https://perma.cc/JQ6G-KSRT)

19.6 Conclusion

In summary, fantasy football is similar to stock picking. You are most likely to pick the best players if you go with the wisdom of the crowd (e.g., average projections across projection sources) and diversify. Most projections are public information, so you might wonder whether using crowd projections gains you anything because everybody else has access to public information. However, this is also the case with stocks, and people still consistently perform best over time when they go with the market. Nevertheless, crowd projections are not highly accurate. And fantasy football is a game, so feel free to have fun and deviate from the crowd! However, you may be just as (if not more) likely to be wrong by deviating from the crowd.

20

Cluster Analysis

20.1 Getting Started

20.1.1 Load Packages

```
library("petersenlab")
library("nflreadr")
library("mclust")
library("plotly")
library("tidyverse")
```

20.1.2 Load Data

```
load(file = "./data/nfl_players.RData")
load(file = "./data/nfl_combine.RData")
load(file = "./data/player_stats_weekly.RData")
load(file = "./data/player_stats_seasonal.RData")
load(file = "./data/nfl_advancedStatsPFR_seasonal.RData")
load(file = "./data/nfl_actualStats_career.RData")
```

20.1.3 Overview

Whereas factor analysis evaluates how *variables* do or do not hang together—in terms of their associations and non-associations, cluster analysis evaluates how *people* are or or not similar—in terms of their scores on one or more variables. The goal of cluster analysis is to identify distinguishable subgroups of people. The people within a subgroup are expected to be more similar to each other than they are to people in other subgroups. For instance, we might expect that there are distinguishable subtypes of Wide Receivers: possession, deep threats, and slot-type Wide Receivers. Possession Wide Receivers tend to

be taller and heavier, with good hands who catch the ball at a high rate. Deep threat Wide Receivers tend to be fast. Slot-type Wide Receivers tend to be small, quick, and agile. In order to identify these clusters of Wide Receivers, we might conduct a cluster analysis with variables relating to the players' height, weight, percent of (catchable) targets caught, air yards received, and various metrics from the National Football League (NFL) Combine, including their times in the 40-yard dash, 20-yard shuttle run, and three cone drill.

There are many approaches to cluster analysis, including model-based clustering, density-based clustering, centroid-based clustering, hierarchical clustering (aka connectivity-based clustering), etc. An overview of approaches to cluster analysis in `R` is provided by Kassambara (2017). In this chapter, we focus on examples using model-based clustering with the `mclust` package (Scrucca et al., 2023), which uses Gaussian finite mixture modeling. The various types of `mclust` models are provided here: https://mclust-org.github.io/mclust/reference/mclustModelNames.html.

20.1.4 Tiers of Prior Season Fantasy Points

20.1.4.1 Prepare Data

```
recentSeason <- max(player_stats_seasonal_offense$season, na.rm = TRUE) # also works: nflreadr::mo
recentSeason
```

```
[1] 2023
```

```
player_stats_seasonal_offense_recent <- player_stats_seasonal_offense %>%
  filter(season == recentSeason)

player_stats_seasonal_offense_recentQB <- player_stats_seasonal_offense_recent %>%
  filter(position_group == "QB")

player_stats_seasonal_offense_recentRB <- player_stats_seasonal_offense_recent %>%
  filter(position_group == "RB")

player_stats_seasonal_offense_recentWR <- player_stats_seasonal_offense_recent %>%
  filter(position_group == "WR")

player_stats_seasonal_offense_recentTE <- player_stats_seasonal_offense_recent %>%
  filter(position_group == "TE")
```

20.1.4.2 Identify the Optimal Number of Tiers by Position

20.1.4.2.1 Quarterbacks

```
tiersQB_bic <- mclust::mclustBIC(
  data = player_stats_seasonal_offense_recentQB$fantasy_points,
  G = 1:9
)

tiersQB_bic
```

```
Bayesian Information Criterion (BIC):
           E          V
1 -1029.2788 -1029.2788
2  -999.8644  -971.9776
3 -1008.6981  -967.2602
4 -1008.7235  -960.6004
5 -1017.5645  -968.2976
6 -1012.3338  -979.2719
7 -1019.6165  -981.8370
8 -1028.4185  -994.2312
9 -1037.2919  -996.8274

Top 3 models based on the BIC criterion:
      V,4       V,3       V,5
-960.6004 -967.2602 -968.2976
```

```
summary(tiersQB_bic)
```

```
Best BIC values:
               V,4         V,3         V,5
BIC      -960.6004 -967.260221 -968.297644
BIC diff    0.0000   -6.659795   -7.697218
```

```
plot(tiersQB_bic)
```

```r
tiersQB_icl <- mclust::mclustICL(
  data = player_stats_seasonal_offense_recentQB$fantasy_points,
  G = 1:9
)

tiersQB_icl
```

```
Integrated Complete-data Likelihood (ICL) criterion:
          E          V
1 -1029.279 -1029.2788
2 -1003.141  -978.2956
3 -1088.651  -979.5620
4 -1090.494  -973.6231
5 -1133.145  -980.2601
6 -1115.235  -991.2491
7 -1114.227  -995.1388
8 -1135.395 -1010.5630
9 -1159.734 -1013.9355

Top 3 models based on the ICL criterion:
      V,4       V,2       V,3
-973.6231 -978.2956 -979.5620
```

```r
summary(tiersQB_icl)
```

```
Best ICL values:
```

```
               V,4         V,2         V,3
ICL      -973.6231 -978.295620 -979.561997
ICL diff    0.0000   -4.672521   -5.938898
```

```r
plot(tiersQB_icl)
```

```r
tiersQB_boostrap <- mclust::mclustBootstrapLRT(
  data = player_stats_seasonal_offense_recentQB$fantasy_points,
  modelName = "V") # variable/unequal variance (for univariate data)

numTiersQB <- as.numeric(summary(tiersQB_boostrap)[,"Length"][1]) # or could specify the number of

tiersQB_boostrap
```

```
--------------------------------------------------------------
Bootstrap sequential LRT for the number of mixture components
--------------------------------------------------------------
Model        = V
Replications = 999
              LRTS bootstrap p-value
1 vs 2   70.521368             0.001
2 vs 3   17.937493             0.001
3 vs 4   19.879953             0.002
4 vs 5    5.522939             0.284
```

```r
plot(
  tiersQB_boostrap,
  G = numTiersQB - 1)
```

20.1.4.2.2 Running Backs

```r
tiersRB_bic <- mclust::mclustBIC(
  data = player_stats_seasonal_offense_recentRB$fantasy_points,
  G = 1:9
)

tiersRB_bic
```

```
Bayesian Information Criterion (BIC):
          E         V
1 -1721.012 -1721.012
2 -1664.673 -1596.519
3 -1674.688 -1559.835
4 -1684.702 -1562.777
5 -1678.476 -1562.212
6 -1688.478        NA
7 -1694.344        NA
8 -1704.351        NA
```

```
9 -1714.393 -1614.038

Top 3 models based on the BIC criterion:
      V,3       V,5       V,4
-1559.835 -1562.212 -1562.777
```

```r
summary(tiersRB_bic)
```

```
Best BIC values:
               V,3          V,5          V,4
BIC      -1559.835 -1562.211682 -1562.776909
BIC diff     0.000    -2.376539    -2.941765
```

```r
plot(tiersRB_bic)
```

```r
tiersRB_icl <- mclust::mclustICL(
  data = player_stats_seasonal_offense_recentRB$fantasy_points,
  G = 1:9
)

tiersRB_icl
```

```
Integrated Complete-data Likelihood (ICL) criterion:
          E         V
1 -1721.012 -1721.012
```

```
2 -1670.491 -1617.112
3 -1825.647 -1585.838
4 -1889.919 -1609.826
5 -1915.554 -1595.334
6 -1974.819        NA
7 -1955.175        NA
8 -1997.059        NA
9 -2042.572 -1680.602

Top 3 models based on the ICL criterion:
      V,3       V,5       V,4
-1585.838 -1595.334 -1609.826
```

```r
summary(tiersRB_icl)
```

```
Best ICL values:
               V,3          V,5         V,4
ICL      -1585.838 -1595.333586 -1609.82628
ICL diff     0.000    -9.495921   -23.98861
```

```r
plot(tiersRB_icl)
```

```r
numTiersRB <- 3
```

The model-based bootstrap clustering of Running Backs' fantasy points is unable to run due to an error:

```
tiersRB_boostrap <- mclust::mclustBootstrapLRT(
  data = player_stats_seasonal_offense_recentRB$fantasy_points,
  modelName = "V") # variable/unequal variance (for univariate data)
```

```
Error in obsLRTS[g] <- 2 * (Mod1$loglik - Mod0$loglik): replacement has length zero
```

Thus, we cannot use the following code, which would otherwise summarize the model results, specify the number of tiers, and plot model comparisons:

```
numTiersRB <- as.numeric(summary(tiersRB_boostrap)[,"Length"][1]) # or could specify the number of

tiersRB_boostrap
plot(
  tiersRB_boostrap,
  G = numTiersRB - 1)
```

20.1.4.2.3 Wide Receivers

```
tiersWR_bic <- mclust::mclustBIC(
  data = player_stats_seasonal_offense_recentWR$fantasy_points,
  G = 1:9
)

tiersWR_bic
```

```
Bayesian Information Criterion (BIC):
          E         V
1 -2370.375 -2370.375
2 -2304.375 -2255.621
3 -2315.140 -2225.899
4 -2299.198 -2219.074
5 -2309.917 -2223.939
6 -2320.651 -2235.922
7 -2305.267 -2248.569
8 -2315.650 -2238.168
9 -2316.998 -2247.356

Top 3 models based on the BIC criterion:
      V,4       V,5       V,3
-2219.074 -2223.939 -2225.899
```

```r
summary(tiersWR_bic)
```

```
Best BIC values:
               V,4          V,5          V,3
BIC      -2219.074 -2223.939239 -2225.898872
BIC diff     0.000    -4.865631    -6.825264
```

```r
plot(tiersWR_bic)
```

```r
tiersWR_icl <- mclust::mclustICL(
  data = player_stats_seasonal_offense_recentWR$fantasy_points,
  G = 1:9
)

tiersWR_icl
```

```
Integrated Complete-data Likelihood (ICL) criterion:
          E         V
1 -2370.375 -2370.375
2 -2316.917 -2300.187
3 -2533.121 -2278.587
4 -2518.776 -2297.528
5 -2642.585 -2311.853
6 -2725.921 -2310.061
```

```
7 -2655.701 -2335.697
8 -2708.675 -2315.514
9 -2713.387 -2303.353

Top 3 models based on the ICL criterion:
      V,3       V,4       V,2
-2278.587 -2297.528 -2300.187
```

```r
summary(tiersWR_icl)
```

```
Best ICL values:
               V,3         V,4         V,2
ICL      -2278.587 -2297.52802 -2300.18723
ICL diff     0.000   -18.94108   -21.60028
```

```r
plot(tiersWR_icl)
```

```r
tiersWR_boostrap <- mclust::mclustBootstrapLRT(
  data = player_stats_seasonal_offense_recentWR$fantasy_points,
  modelName = "V") # variable/unequal variance (for univariate data)

numTiersWR <- as.numeric(summary(tiersWR_boostrap)[,"Length"][1]) # or could specify the number of

tiersWR_boostrap
```

```
---------------------------------------------------------------
Bootstrap sequential LRT for the number of mixture components
---------------------------------------------------------------
Model        = V
Replications = 999
               LRTS bootstrap p-value
1 vs 2   130.851395             0.001
2 vs 3    45.820182             0.001
3 vs 4    22.923192             0.001
4 vs 5    11.232297             0.021
5 vs 6     4.114727             0.260
```

```r
plot(
  tiersWR_boostrap,
  G = numTiersWR - 1)
```

20.1.4.2.4 Tight Ends

```r
tiersTE_bic <- mclust::mclustBIC(
  data = player_stats_seasonal_offense_recentTE$fantasy_points,
  G = 1:9
)

tiersTE_bic
```

```
Bayesian Information Criterion (BIC):
          E         V
1 -1191.234 -1191.234
2 -1154.104 -1123.349
3 -1163.614 -1105.110
4 -1173.135 -1107.427
5 -1162.852 -1116.659
6 -1172.360 -1125.435
7 -1166.107 -1136.896
8 -1175.610 -1144.981
9 -1172.515 -1157.098

Top 3 models based on the BIC criterion:
      V,3       V,4       V,5
-1105.110 -1107.427 -1116.659
```

```r
summary(tiersTE_bic)
```

```
Best BIC values:
              V,3          V,4         V,5
BIC      -1105.11 -1107.427151 -1116.65944
BIC diff     0.00    -2.317554   -11.54984
```

```r
plot(tiersTE_bic)
```

```r
tiersTE_icl <- mclust::mclustICL(
  data = player_stats_seasonal_offense_recentTE$fantasy_points,
  G = 1:9
)

tiersTE_icl
```

```
Integrated Complete-data Likelihood (ICL) criterion:
          E         V
1 -1191.234 -1191.234
2 -1159.663 -1146.435
3 -1292.566 -1136.471
4 -1375.784 -1153.849
5 -1349.189 -1159.467
6 -1403.580 -1159.801
7 -1368.315 -1178.065
8 -1406.376 -1169.766
9 -1401.273 -1175.813

Top 3 models based on the ICL criterion:
      V,3       V,2       V,4
-1136.471 -1146.435 -1153.849
```

```r
summary(tiersTE_icl)
```

```
Best ICL values:
               V,3          V,2       V,4
ICL      -1136.471 -1146.435247 -1153.849
ICL diff     0.000    -9.963956   -17.378
```

```r
plot(tiersTE_icl)
```

```
tiersTE_boostrap <- mclust::mclustBootstrapLRT(
  data = player_stats_seasonal_offense_recentTE$fantasy_points,
  modelName = "V") # variable/unequal variance (for univariate data)

numTiersTE <- as.numeric(summary(tiersTE_boostrap)[,"Length"][1]) # or could specify the number of

tiersTE_boostrap
```

```
--------------------------------------------------------------
Bootstrap sequential LRT for the number of mixture components
--------------------------------------------------------------
Model        = V
Replications = 999
              LRTS bootstrap p-value
1 vs 2   82.145958             0.001
2 vs 3   32.500344             0.001
3 vs 4   11.943217             0.017
4 vs 5    5.028484             0.262
```

```
plot(
  tiersTE_boostrap,
  G = numTiersTE - 1)
```

20.1.4.3 Fit the Cluster Model to the Optimal Number of Tiers

20.1.4.3.1 Quarterbacks

In our data, all of the following models are equivalent—i.e., they result in the same unequal variance model with a 4-cluster solution—but they arrive there in different ways.

```
mclust::Mclust(
  data = player_stats_seasonal_offense_recentQB$fantasy_points,
  G = numTiersQB,
)

mclust::Mclust(
  data = player_stats_seasonal_offense_recentQB$fantasy_points,
  G = 4,
)

mclust::Mclust(
  data = player_stats_seasonal_offense_recentQB$fantasy_points,
)

mclust::Mclust(
  data = player_stats_seasonal_offense_recentQB$fantasy_points,
```

```
  x = tiersQB_bic
)
```

Let's fit one of these:

```
clusterModelQBs <- mclust::Mclust(
  data = player_stats_seasonal_offense_recentQB$fantasy_points,
  G = numTiersQB,
)
```

Here are the number of players that are in each of the four clusters (i.e., tiers):

```
table(clusterModelQBs$classification)
```

```
 1  2  3  4
13 21 26 22
```

20.1.4.3.2 Running Backs

```
clusterModelRBs <- mclust::Mclust(
  data = player_stats_seasonal_offense_recentRB$fantasy_points,
  G = numTiersRB,
)
```

Here are the number of players that are in each of the four clusters (i.e., tiers):

```
table(clusterModelRBs$classification)
```

```
 1  2  3
32 55 62
```

20.1.4.3.3 Wide Receivers

```
clusterModelWRs <- mclust::Mclust(
  data = player_stats_seasonal_offense_recentWR$fantasy_points,
  G = numTiersWR,
)
```

Here are the number of players that are in each of the four clusters (i.e., tiers):

```
table(clusterModelWRs$classification)
```

```
 1  2  3  4  5 
24 35 62 42 51 
```

20.1.4.3.4 Tight Ends

```
clusterModelTEs <- mclust::Mclust(
  data = player_stats_seasonal_offense_recentTE$fantasy_points,
  G = numTiersTE,
)
```

Here are the number of players that are in each of the four clusters (i.e., tiers):

```
table(clusterModelTEs$classification)
```

```
 1  2  3  4 
27 38 32 19 
```

20.1.4.4 Plot the Tiers

We can merge the player's classification into the dataset and plot each player's classification.

20.1.4.4.1 Quarterbacks

```
player_stats_seasonal_offense_recentQB$tier <- clusterModelQBs$classification

player_stats_seasonal_offense_recentQB <- player_stats_seasonal_offense_recentQB %>%
  mutate(
    tier = factor(case_match( # recode so 1 is highest tier
      tier,
      1 ~ 4,
      2 ~ 3,
```

```
      3 ~ 2,
      4 ~ 1)))

player_stats_seasonal_offense_recentQB$position_rank <- rank(
  player_stats_seasonal_offense_recentQB$fantasy_points * -1,
  na.last = "keep",
  ties.method = "min")

plot_qbTiers <- ggplot2::ggplot(
  data = player_stats_seasonal_offense_recentQB,
  mapping = aes(
    x = fantasy_points,
    y = position_rank,
    color = tier
  )) +
  geom_point(
    aes(
      text = player_display_name # add player name for mouse over tooltip
  )) +
  scale_y_continuous(trans = "reverse") +
  coord_cartesian(clip = "off") +
  labs(
    x = "Projected Points",
    y = "Position Rank",
    title = "Quarterback Fantasy Points by Tier",
    color = "Tier") +
  theme_classic() +
  theme(legend.position = "top")

ggplotly(plot_qbTiers)
```

Figure 20.1 Quarterback Fantasy Points by Tier.

20.1.4.4.2 Running Backs

```
player_stats_seasonal_offense_recentRB$tier <- clusterModelRBs$classification

player_stats_seasonal_offense_recentRB <- player_stats_seasonal_offense_recentRB %>%
  mutate(
    tier = factor(case_match( # recode so 1 is highest tier
      tier,
      1 ~ 3,
      2 ~ 2,
      3 ~ 1)))

player_stats_seasonal_offense_recentRB$position_rank <- rank(
  player_stats_seasonal_offense_recentRB$fantasy_points * -1,
  na.last = "keep",
  ties.method = "min")

plot_rbTiers <- ggplot2::ggplot(
  data = player_stats_seasonal_offense_recentRB,
  mapping = aes(
    x = fantasy_points,
    y = position_rank,
    color = tier
  )) +
  geom_point(
    aes(
      text = player_display_name # add player name for mouse over tooltip
  )) +
  scale_y_continuous(trans = "reverse") +
  coord_cartesian(clip = "off") +
  labs(
    x = "Projected Points",
    y = "Position Rank",
    title = "Running Back Fantasy Points by Tier",
    color = "Tier") +
  theme_classic() +
  theme(legend.position = "top")

ggplotly(plot_rbTiers)
```

Figure 20.2 Running Back Fantasy Points by Tier.

20.1.4.4.3 Wide Receivers

```
player_stats_seasonal_offense_recentWR$tier <- clusterModelWRs$classification

player_stats_seasonal_offense_recentWR <- player_stats_seasonal_offense_recentWR %>%
  mutate(
    tier = factor(case_match( # recode so 1 is highest tier
      tier,
      1 ~ 5,
      2 ~ 4,
      3 ~ 3,
      4 ~ 2,
      5 ~ 1)))

player_stats_seasonal_offense_recentWR$position_rank <- rank(
  player_stats_seasonal_offense_recentWR$fantasy_points * -1,
  na.last = "keep",
  ties.method = "min")

plot_wrTiers <- ggplot2::ggplot(
  data = player_stats_seasonal_offense_recentWR,
  mapping = aes(
    x = fantasy_points,
    y = position_rank,
    color = tier
  )) +
  geom_point(
    aes(
      text = player_display_name # add player name for mouse over tooltip
  )) +
  scale_y_continuous(trans = "reverse") +
  coord_cartesian(clip = "off") +
  labs(
    x = "Projected Points",
    y = "Position Rank",
    title = "Wide Receiver Fantasy Points by Tier",
    color = "Tier") +
  theme_classic() +
  theme(legend.position = "top")

ggplotly(plot_wrTiers)
```

Figure 20.3 Quarterback Fantasy Points by Tier.

20.1.4.4.4 Tight Ends

```
player_stats_seasonal_offense_recentTE$tier <- clusterModelTEs$classification

player_stats_seasonal_offense_recentTE <- player_stats_seasonal_offense_recentTE %>%
  mutate(
    tier = factor(case_match( # recode so 1 is highest tier
      tier,
      1 ~ 4,
      2 ~ 3,
      3 ~ 2,
      4 ~ 1)))

player_stats_seasonal_offense_recentTE$position_rank <- rank(
  player_stats_seasonal_offense_recentTE$fantasy_points * -1,
  na.last = "keep",
  ties.method = "min")

plot_teTiers <- ggplot2::ggplot(
  data = player_stats_seasonal_offense_recentTE,
  mapping = aes(
    x = fantasy_points,
    y = position_rank,
    color = tier
  )) +
  geom_point(
    aes(
      text = player_display_name # add player name for mouse over tooltip
  )) +
  scale_y_continuous(trans = "reverse") +
  coord_cartesian(clip = "off") +
  labs(
    x = "Projected Points",
    y = "Position Rank",
    title = "Tight End Fantasy Points by Tier",
    color = "Tier") +
  theme_classic() +
  theme(legend.position = "top")

ggplotly(plot_teTiers)
```

Figure 20.4 Tight End Fantasy Points by Tier.

20.1.5 Types of Wide Receivers

```
# Compute Advanced PFR Stats by Career
pfrVars <- nfl_advancedStatsPFR_seasonal %>%
```

```
  select(pocket_time.pass:cmp_percent.def, g, gs) %>%
  names()

weightedAverageVars <- c(
  "pocket_time.pass",
  "ybc_att.rush","yac_att.rush",
  "ybc_r.rec","yac_r.rec","adot.rec","rat.rec",
  "yds_cmp.def","yds_tgt.def","dadot.def","m_tkl_percent.def","rat.def"
)

recomputeVars <- c(
  "drop_pct.pass", # drops.pass / pass_attempts.pass
  "bad_throw_pct.pass", # bad_throws.pass / pass_attempts.pass
  "on_tgt_pct.pass", # on_tgt_throws.pass / pass_attempts.pass
  "pressure_pct.pass", # times_pressured.pass / pass_attempts.pass
  "drop_percent.rec", # drop.rec / tgt.rec
  "rec_br.rec", # rec.rec / brk_tkl.rec
  "cmp_percent.def" # cmp.def / tgt.def
)

sumVars <- pfrVars[pfrVars %ni% c(
  weightedAverageVars, recomputeVars,
  "merge_name", "loaded.pass", "loaded.rush", "loaded.rec", "loaded.def")]

nfl_advancedStatsPFR_career <- nfl_advancedStatsPFR_seasonal %>%
  group_by(pfr_id, merge_name) %>%
  summarise(
    across(all_of(weightedAverageVars), ~ weighted.mean(.x, w = g, na.rm = TRUE)),
    across(all_of(sumVars), ~ sum(.x, na.rm = TRUE)),
    .groups = "drop") %>%
  mutate(
    drop_pct.pass = drops.pass / pass_attempts.pass,
    bad_throw_pct.pass = bad_throws.pass / pass_attempts.pass,
    on_tgt_pct.pass = on_tgt_throws.pass / pass_attempts.pass,
    pressure_pct.pass = times_pressured.pass / pass_attempts.pass,
    drop_percent.rec = drop.rec / tgt.rec,
    rec_br.rec = drop.rec / tgt.rec,
    cmp_percent.def = cmp.def / tgt.def
  )

uniqueCases <- nfl_advancedStatsPFR_seasonal %>% select(pfr_id, merge_name, gsis_id) %>% unique()

uniqueCases %>%
  group_by(pfr_id) %>%
```

```
  filter(n() > 1)
```

```
# A tibble: 6 x 3
# Groups:   pfr_id [3]
  pfr_id   merge_name      gsis_id
  <chr>    <chr>           <chr>
1 AndeRo04 chosen anderson 00-0032688
2 AndeRo04 robbie chosen   00-0032688
3 WillMa06 marcus williams <NA>
4 WillMa06 marcus williams 00-0033894
5 WoolTa00 riq woolen      00-0037079
6 WoolTa00 tariq woolen    00-0037079
```

```
nfl_advancedStatsPFR_seasonal <- nfl_advancedStatsPFR_seasonal %>%
  filter(pfr_id != "WillMa06" | merge_name != "MARCUSWILLIAMS" | !is.na(gsis_id))


nfl_advancedStatsPFR_career <- left_join(
  nfl_advancedStatsPFR_career,
  nfl_advancedStatsPFR_seasonal %>% select(pfr_id, merge_name, gsis_id) %>% unique(),
  by = c("pfr_id", "merge_name")
)

# Compute Player Stats Per Season
player_stats_seasonal_careerWRs <- player_stats_seasonal_offense %>%
  filter(position == "WR") %>%
  group_by(player_id) %>%
  summarise(
    across(all_of(c("targets", "receptions", "receiving_air_yards")), ~ weighted.mean(.x, w = game
    .groups = "drop")

# Drop players with no receiving air yards
player_stats_seasonal_careerWRs <- player_stats_seasonal_careerWRs %>%
  filter(receiving_air_yards != 0) %>%
  rename(
    targets_per_season = targets,
    receptions_per_season = receptions,
    receiving_air_yards_per_season = receiving_air_yards
  )

# Merge
playerListToMerge <- list(
  nfl_players %>% select(gsis_id, display_name, position, height, weight),
  nfl_combine %>% select(gsis_id, vertical, forty, ht, wt),
```

```
  player_stats_seasonal_careerWRs %>% select(player_id, targets_per_season, receptions_per_season,
    rename(gsis_id = player_id),
  nfl_actualStats_offense_career %>% select(player_id, receptions, targets, receiving_air_yards, a
    rename(gsis_id = player_id),
  nfl_advancedStatsPFR_career %>% select(gsis_id, adot.rec, rec.rec, brk_tkl.rec, drop.rec, drop_p
)

merged_data <- playerListToMerge %>%
  reduce(
    full_join,
    by = c("gsis_id"),
    na_matches = "never")
```

Additional processing:

```
merged_data <- merged_data %>%
  mutate(
    height_coalesced = coalesce(height, ht),
    weight_coalesced = coalesce(weight, wt),
    receptions_coalesced = pmax(receptions, rec.rec, na.rm = TRUE),
    receiving_air_yards_per_rec = receiving_air_yards / receptions
  )

merged_data$receiving_air_yards_per_rec[which(merged_data$receptions == 0)] <- 0

merged_dataWRs <- merged_data %>%
  filter(position == "WR")

merged_dataWRs_cluster <- merged_dataWRs %>%
  filter(receptions_coalesced >= 100) %>% # keep WRs with at least 100 receptions
  select(gsis_id, display_name, vertical, forty, height_coalesced, weight_coalesced, adot.rec, dro
  na.omit()
```

20.1.5.1 Identify the Number of WR Types

```
wrTypes_bic <- mclust::mclustBIC(
  data = merged_dataWRs_cluster %>% select(-gsis_id, -display_name),
  G = 1:9
)

wrTypes_bic
```

```
Bayesian Information Criterion (BIC):
```

```
        EII       VII       EEI       VEI       EVI       VVI       EEE
1 -7621.294 -7621.294 -4614.720 -4614.720 -4614.720 -4614.720 -4473.015
2 -7169.871 -7156.868 -4583.609 -4593.523 -4474.969 -4592.005 -4494.911
3 -7094.619 -7049.055 -4546.711 -4550.701 -4476.815 -4451.189 -4500.037
4 -6994.734 -6983.329 -4544.846 -4507.683 -4449.172 -4463.329 -4468.862
5 -6878.432 -6883.667 -4546.040 -4506.485 -4487.491 -4475.100 -4488.615
6 -6873.055 -6853.093 -4551.870 -4509.194 -4527.513 -4504.155 -4527.696
7 -6863.284 -6851.272 -4572.778 -4536.946 -4568.859 -4545.793 -4546.801
8 -6883.208 -6830.273 -4596.929 -4563.766 -4615.636 -4600.629 -4532.842
9 -6868.511 -6795.753 -4608.534 -4572.191        NA -4674.032 -4577.602
        VEE       EVE       VVE       EEV       VEV       EVV       VVV
1 -4473.015 -4473.015 -4473.015 -4473.015 -4473.015 -4473.015 -4473.015
2 -4486.812 -4263.868 -4495.504 -4433.428 -4333.476 -4421.039 -4425.529
3 -4493.810 -4241.056 -4251.481 -4461.695 -4449.506 -4512.485 -4511.344
4 -4413.662 -4259.470 -4280.065 -4495.213 -4495.499 -4567.542 -4554.161
5 -4455.285 -4339.280 -4326.608 -4677.378 -4752.961        NA        NA
6        NA        NA        NA -4750.346 -4809.005        NA        NA
7        NA        NA        NA -4921.423 -4870.526        NA        NA
8        NA        NA        NA -5017.409 -4949.576        NA        NA
9        NA        NA        NA -5194.657 -5056.705        NA        NA

Top 3 models based on the BIC criterion:
    EVE,3     VVE,3     EVE,4
-4241.056 -4251.481 -4259.470
```

```r
summary(wrTypes_bic)
```

```
Best BIC values:
             EVE,3       VVE,3       EVE,4
BIC      -4241.056 -4251.48115 -4259.47030
BIC diff     0.000   -10.42502   -18.41417
```

```r
plot(wrTypes_bic)
```

```
wrTypes_icl <- mclust::mclustICL(
  data = merged_dataWRs_cluster %>% select(-gsis_id, -display_name),
  G = 1:9
)

wrTypes_icl
```

```
Integrated Complete-data Likelihood (ICL) criterion:
        EII       VII       EEI       VEI       EVI       VVI       EEE
1 -7621.294 -7621.294 -4614.720 -4614.720 -4614.720 -4614.720 -4473.015
2 -7173.580 -7160.875 -4604.851 -4614.071 -4487.919 -4616.891 -4502.693
3 -7108.034 -7060.233 -4566.469 -4569.834 -4490.373 -4467.963 -4511.260
4 -7005.172 -6992.669 -4564.093 -4523.721 -4459.266 -4479.389 -4476.863
5 -6887.015 -6890.199 -4566.105 -4520.616 -4497.621 -4487.156 -4497.046
6 -6883.359 -6859.414 -4571.773 -4522.768 -4541.911 -4517.646 -4541.667
7 -6874.353 -6857.226 -4596.590 -4552.992 -4584.905 -4555.157 -4559.105
8 -6894.588 -6836.433 -4614.985 -4580.111 -4628.714 -4608.939 -4550.436
9 -6880.988 -6801.354 -4624.659 -4583.616        NA -4682.602 -4587.531
        VEE       EVE       VVE       EEV       VEV       EVV       VVV
1 -4473.015 -4473.015 -4473.015 -4473.015 -4473.015 -4473.015 -4473.015
2 -4496.343 -4265.941 -4505.658 -4435.818 -4333.594 -4421.937 -4426.353
3 -4506.933 -4253.286 -4267.778 -4462.883 -4450.214 -4515.978 -4514.797
4 -4421.326 -4267.576 -4288.583 -4498.571 -4498.800 -4570.414 -4558.586
5 -4463.365 -4350.329 -4332.542 -4684.627 -4756.239        NA        NA
6        NA        NA        NA -4754.258 -4810.294        NA        NA
7        NA        NA        NA -4923.346 -4870.984        NA        NA
```

```
8        NA        NA        NA -5017.690 -4949.791        NA        NA
9        NA        NA        NA -5194.833 -5056.924        NA        NA

Top 3 models based on the ICL criterion:
    EVE,3     EVE,2     EVE,4
-4253.286 -4265.941 -4267.576
```

```
summary(wrTypes_icl)
```

```
Best ICL values:
             EVE,3       EVE,2       EVE,4
ICL      -4253.286 -4265.94062 -4267.57568
ICL diff     0.000   -12.65428   -14.28934
```

```
plot(wrTypes_icl)
```

Based on the cluster analyses, it appears that three clusters are the best fit to the data.

```
numTypesWR <- 3
```

```
wrTypes_boostrap <- mclust::mclustBootstrapLRT(
  data = merged_dataWRs_cluster %>% select(-gsis_id, -display_name),
  modelName = "EVE") # ellipsoidal with equal volume, variable shape, and equal orientation (for m
```

```
wrTypes_boostrap
plot(
  wrTypes_boostrap,
  G = numTypesWR - 1)
```

20.1.5.2 Fit the Cluster Model to the Optimal Number of WR Types

```
clusterModelWRtypes <- mclust::Mclust(
  data = merged_dataWRs_cluster %>% select(-gsis_id, -display_name),
  G = numTypesWR,
)

summary(clusterModelWRtypes)
```

```
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust EVE (ellipsoidal, equal volume and orientation) model with 3 components:

 log-likelihood   n df       BIC       ICL
      -1907.796 113 90 -4241.056 -4253.286

Clustering table:
 1  2  3
45 55 13
```

20.1.5.3 Plots of the Cluster Model

```
plot(
  clusterModelWRtypes,
  what = "BIC")
```

```
plot(
  clusterModelWRtypes,
  what = "classification")
```

```
plot(
  clusterModelWRtypes,
  what = "uncertainty")
```

```
plot(
  clusterModelWRtypes,
  what = "density")
```

20.1.5.4 Interpreting the Clusters

```
table(clusterModelWRtypes$classification)
```

```
 1  2  3 
45 55 13 
```

```
merged_dataWRs_cluster$type <- clusterModelWRtypes$classification

merged_dataWRs_cluster %>%
  group_by(type) %>%
  summarise(across(
    where(is.numeric),
    ~ mean(., na.rm = TRUE)
    )) %>%
  t() %>%
  round(., 2)
```

```
                              [,1]   [,2]   [,3]
type                          1.00   2.00   3.00
vertical                     36.43  35.92  36.77
forty                         4.47   4.47   4.45
height_coalesced             72.27  73.24  72.85
weight_coalesced            202.18 200.65 205.62
adot.rec                      9.36  11.53  12.14
drop_percent.rec              0.04   0.05   0.07
receiving_air_yards_per_rec  14.13  19.95  22.79
brk_tkl.rec                  19.20   7.27   0.38
receptions_per_season        75.10  44.99  43.92
```

Based on this analysis (and the variables included), there appear to be three types of Wide Receivers. Type 1 Wide Receivers includes the Elite WR1s who are strong possession receivers (note: not all players in a given cluster map on perfectly to the typology—i.e., not all Type 1 Wide Receivers are elite WR1s). They tend to have the lowest drop percentage, the shortest average depth of target, and the fewest receiving air yards per reception. They tend to have the most receptions per season and break the most tackles.

Type 2 Wide Receivers includes the consistent contributor, WR2 types. They had fewer receptions and fewer broken tackles than Type 1 Wide Receivers. Their average depth of target was longer than Type 1, and they had more receiving air yards per reception than Type 1.

Type 3 Wide Receivers includes the deep threats. They have the greatest average depth of target and the most receiving yards per reception. However, they also have the fewest receptions, the highest drop percentage, and the fewest broken tackles. Thus, they may be considered the boom-or-bust Wide Receivers.

The tiers were not particularly distinguishable based on their height, weight, vertical jump, or forty-yard dash time.

Type 1 ("Elite/WR1") WRs:

```
merged_dataWRs_cluster %>%
  filter(type == 1) %>%
  select(display_name)
```

```
# A tibble: 45 x 1
   display_name
   <chr>
 1 A.J. Brown
 2 Amari Cooper
 3 Amon-Ra St. Brown
 4 Antonio Brown
 5 Brandin Cooks
 6 Brandon Aiyuk
 7 CeeDee Lamb
 8 Chris Godwin
 9 Christian Kirk
10 Cooper Kupp
# i 35 more rows
```

Type 2 ("Consistent Contributor/WR2") WRs:

```
merged_dataWRs_cluster %>%
  filter(type == 2) %>%
  select(display_name)
```

```
# A tibble: 55 x 1
   display_name
   <chr>
 1 A.J. Green
 2 Albert Wilson
 3 Allen Lazard
 4 Allen Robinson
 5 Byron Pringle
 6 Calvin Ridley
```

```
 7 Cedrick Wilson
 8 Chase Claypool
 9 Chris Conley
10 Chris Moore
# i 45 more rows
```

Type 3 ("Deep Threat/Boom-or-Bust") WRs:

```r
merged_dataWRs_cluster %>%
  filter(type == 3) %>%
  select(display_name)
```

```
# A tibble: 13 x 1
   display_name
   <chr>
 1 Andre Roberts
 2 Brandon LaFell
 3 Brandon Marshall
 4 Brian Quick
 5 Darrius Heyward-Bey
 6 Jeremy Kerley
 7 Marqise Lee
 8 Michael Floyd
 9 Mike Wallace
10 Rishard Matthews
11 Terrance Williams
12 Torrey Smith
13 Travis Benjamin
```

20.2 Conclusion

21

Time Series Analysis

21.1 Getting Started

21.1.1 Load Packages

```
library("petersenlab")
library("xts")
library("zoo")
library("forecast")
library("brms")
library("plotly")
library("tidyverse")
```

21.1.2 Load Data

```
load(file = "./data/player_stats_weekly.RData")
load(file = "./data/player_stats_seasonal.RData")
load(url("https://osf.io/download/q6rjf/"))
```

21.2 Overview of Time Series Analysis

Time series analysis is useful when trying to generate forecasts from longitudinal data. That is, time series analysis seeks to evaluate change over time to predict future values.

There many different types of time series analyses. For simplicity, in this chapter, we use autoregressive integrated moving average (ARIMA) models

to demonstrate one approach to time series analysis. We also leverage Bayesian mixed models to generate forecasts of future performance and plots of individuals model-implied performance by age and position.

21.3 Autoregressive Integrated Moving Average (ARIMA) Models

Hyndman & Athanasopoulos (2021) provide a nice overview of ARIMA models. As noted by Hyndman & Athanasopoulos (2021), ARIMA models aim to describe how a variable is correlated with itself over time (autocorrelation)—i.e., how earlier levels of a variable are correlated with later levels of the same variable. ARIMA models perform best when there is a clear pattern where later values are influenced by earlier values. ARIMA models incorporate autoregression effects, moving average effects, and differencing.

ARIMA models can have various numbers of terms and model complexity. They are specified in the following form: ARIMA(p, d, q), where:

- $p =$ the number of autoregressive terms
- $d =$ the number of differences between consecutive scores (to make the time series stationary by reducing trends and seasonality)
- $q =$ the number of moving average terms

ARIMA models assume that the data are stationary (i.e., there are no long-term trends), are non-seasonal (i.e., there is no consistency of the timing of the peaks or troughs in the line), and that earlier values influence later values. This may not strongly be the case in fantasy football, so ARIMA models may not be particularly useful in forecasting fantasy football performance. Other approaches, such as exponential smoothing, may be useful for data that show longer-term trends and seasonality (Hyndman & Athanasopoulos, 2021). Nevertheless, ARIMA models are widely used in forecasting financial markets and economic indicators. Thus, it is a useful technique to learn.

Adapted from: https://rc2e.com/timeseriesanalysis (archived at https://perma.cc/U5P6-2VWC).

21.3.1 Create the Time Series Objects

```
weeklyFantasyPoints_tomBrady <- player_stats_weekly_offense %>%
  filter(
    player_id == "00-0019596" | player_display_name == "Tom Brady")

weeklyFantasyPoints_peytonManning <- player_stats_weekly_offense %>%
  filter(
    player_id == "00-0010346" | player_display_name == "Peyton Manning")

ts_tomBrady <- xts::xts(
  x = weeklyFantasyPoints_tomBrady["fantasy_points"],
  order.by = weeklyFantasyPoints_tomBrady$gameday)

ts_peytonManning <- xts::xts(
  x = weeklyFantasyPoints_peytonManning["fantasy_points"],
  order.by = weeklyFantasyPoints_peytonManning$gameday)

ts_tomBrady
```

```
           fantasy_points
2000-11-23           0.24
2001-09-23           2.74
2001-09-30           6.92
2001-10-07           0.34
2001-10-14          22.56
2001-10-21          19.88
2001-10-28           8.02
2001-11-04          22.00
2001-11-11           4.18
2001-11-18           8.00
       ...
2022-11-06          15.20
2022-11-13          16.02
2022-11-27          18.04
2022-12-05          17.14
2022-12-11          10.12
2022-12-18          16.58
2022-12-25          11.34
2023-01-01          37.68
2023-01-08           7.36
2023-01-16          22.04
```

```
ts_peytonManning
```

```
           fantasy_points
```

```
1999-09-12          15.06
1999-09-19          16.40
1999-09-26          29.56
1999-10-10          19.66
1999-10-17          10.10
1999-10-24          17.86
1999-10-31          17.76
1999-11-07          17.76
1999-11-14          15.18
1999-11-21          22.00
       ...
2015-10-04           8.32
2015-10-11           6.64
2015-10-18           9.60
2015-11-01          11.60
2015-11-08          15.24
2015-11-15          -6.60
2016-01-03           2.56
2016-01-17          10.78
2016-01-24          14.14
2016-02-07           3.64
```

```r
ts_combined <- merge(
  ts_tomBrady,
  ts_peytonManning
)

names(ts_combined) <- c("Tom Brady","Peyton Manning")
```

21.3.2 Plot the Time Series

```r
plot(
  ts_tomBrady,
  main = "Tom Brady's Fantasy Points by Game")
```

Figure 21.1 Tom Brady's Historical Fantasy Points by Game.

```
plot(
  ts_combined,
  legend,
  legend.loc = "topright",
  main = "Fantasy Points by Game")
```

Figure 21.2 Historical Fantasy Points by Game for Tom Brady and Peyone Manning.

21.3.3 Rolling Mean/Median

```
zoo::rollmean(
  x = ts_tomBrady,
  k = 5)
```

```
           fantasy_points
2001-09-30          6.560
2001-10-07         10.488
2001-10-14         11.544
2001-10-21         14.560
2001-10-28         15.328
2001-11-04         12.416
2001-11-11         13.984
2001-11-18         14.084
2001-11-25         10.568
2001-12-02         11.488
       ...
2022-10-23         15.492
2022-10-27         14.748
2022-11-06         15.612
```

```
2022-11-13         16.700
2022-11-27         15.304
2022-12-05         15.580
2022-12-11         14.644
2022-12-18         18.572
2022-12-25         16.616
2023-01-01         19.000
```

```
zoo::rollmedian(
  x = ts_tomBrady,
  k = 5)
```

```
           fantasy_points
2001-09-30           2.74
2001-10-07           6.92
2001-10-14           8.02
2001-10-21          19.88
2001-10-28          19.88
2001-11-04           8.02
2001-11-11           8.02
2001-11-18           8.52
2001-11-25           8.00
2001-12-02           8.52
       ...
2022-10-23          15.20
2022-10-27          15.20
2022-11-06          16.02
2022-11-13          17.10
2022-11-27          16.02
2022-12-05          16.58
2022-12-11          16.58
2022-12-18          16.58
2022-12-25          11.34
2023-01-01          16.58
```

21.3.4 Autocorrelation

The autocorrelation function (ACF) plot depicts the autocorrelation of scores as a function of the length of the lag. Significant autocorrelation is detected when the autocorrelation exceeds the dashed blue lines, as is depicted in Figure 21.3.

```
acf(ts_tomBrady)
```

Figure 21.3 Autocorrelation Function (ACF) Plot of Tom Brady's Historical Fantasy Points by Game.

```
Box.test(ts_tomBrady)
```

```
    Box-Pierce test

data:  ts_tomBrady
X-squared = 6.9696, df = 1, p-value = 0.008291
```

21.3.5 Fit an Autoregressive Integrated Moving Average Model

```
forecast::auto.arima(ts_tomBrady)
```

```
Series: ts_tomBrady
ARIMA(5,1,4)

Coefficients:
```

```
         ar1      ar2      ar3      ar4     ar5      ma1     ma2      ma3
      0.4519  -0.4600  -0.1781  -0.0834  0.2287  -1.3634  0.8837  -0.1480
s.e.  0.2444   0.2272   0.3634   0.0585  0.0535   0.2504  0.4110   0.5295
          ma4
      -0.3333
s.e.   0.3496

sigma^2 = 59.26:  log likelihood = -1318.42
AIC=2656.83   AICc=2657.42   BIC=2696.29
```

```
forecast::auto.arima(ts_peytonManning)
```

```
Series: ts_peytonManning
ARIMA(1,0,1) with non-zero mean

Coefficients:
         ar1      ma1     mean
      0.9134  -0.8192  17.4587
s.e.  0.0605   0.0786   0.9682

sigma^2 = 60.52:  log likelihood = -959.87
AIC=1927.73   AICc=1927.88   BIC=1942.23
```

```
arima_tomBrady <- arima(
  ts_tomBrady,
  order = c(5, 1, 4))

summary(arima_tomBrady)
```

```
Call:
arima(x = ts_tomBrady, order = c(5, 1, 4))

Coefficients:
         ar1      ar2      ar3      ar4     ar5      ma1     ma2      ma3
      0.4519  -0.4600  -0.1781  -0.0834  0.2287  -1.3634  0.8837  -0.1480
s.e.  0.2444   0.2272   0.3634   0.0585  0.0535   0.2504  0.4110   0.5295
          ma4
      -0.3333
s.e.   0.3496

sigma^2 estimated as 57.86:  log likelihood = -1318.42,  aic = 2656.83

Training set error measures:
```

```
                    ME     RMSE      MAE       MPE     MAPE      MASE
Training set 0.6138177 7.596882 6.059765 -26.43736 54.72413 0.7239184
                     ACF1
Training set -0.005875514
```

```r
confint(arima_tomBrady)
```

```
          2.5 %      97.5 %
ar1 -0.02718585  0.93100467
ar2 -0.90534347 -0.01472881
ar3 -0.89026492  0.53413092
ar4 -0.19813198  0.03125533
ar5  0.12376560  0.33358684
ma1 -1.85414989 -0.87259362
ma2  0.07825054  1.68916699
ma3 -1.18577457  0.88978834
ma4 -1.01847540  0.35186954
```

```r
forecast::checkresiduals(arima_tomBrady)
```

```
    Ljung-Box test

data:  Residuals from ARIMA(5,1,4)
Q* = 2.2811, df = 3, p-value = 0.5162

Model df: 9.   Total lags used: 12
```

Figure 21.4 Model Summary of Autoregressive Integrated Moving Average Model fit to Tom Brady's Historical Performance by Game.

```
arima_tomBrady_removeNonSigTerms <- arima(
  ts_tomBrady,
  order = c(5, 1, 4),
  fixed = c(NA, NA, 0, NA, NA, NA, NA, NA, NA))

summary(arima_tomBrady_removeNonSigTerms)
```

```
Call:
arima(x = ts_tomBrady, order = c(5, 1, 4), fixed = c(NA, NA, 0, NA, NA, NA,
    NA, NA, NA))

Coefficients:
        ar1     ar2  ar3     ar4    ar5     ma1    ma2     ma3     ma4
     0.5195 -0.5537   0 -0.0923 0.2248 -1.4320 1.0411 -0.4109 -0.1649
s.e. 0.2244  0.2095   0  0.0579 0.0559  0.2311 0.3713  0.2145  0.0657

sigma^2 estimated as 57.92:  log likelihood = -1318.59,  aic = 2655.18

Training set error measures:
                    ME     RMSE      MAE       MPE     MAPE      MASE
Training set 0.6222971 7.600691 6.078548 -26.34234 54.91081 0.7261622
                     ACF1
```

```
Training set -0.005570349
```

```r
confint(arima_tomBrady_removeNonSigTerms)
```

```
          2.5 %       97.5 %
ar1  0.07978268  0.959242224
ar2 -0.96433523 -0.143157528
ar3          NA           NA
ar4 -0.20569255  0.021080469
ar5  0.11530044  0.334344203
ma1 -1.88493886 -0.979076737
ma2  0.31341100  1.768874322
ma3 -0.83134627  0.009456031
ma4 -0.29362802 -0.036192104
```

```r
forecast::checkresiduals(arima_tomBrady_removeNonSigTerms)
```

```
    Ljung-Box test

data:  Residuals from ARIMA(5,1,4)
Q* = 2.3773, df = 3, p-value = 0.4979

Model df: 9.   Total lags used: 12
```

Figure 21.5 Model Summary of modified Autoregressive Integrated Moving Average Model fit to Tom Brady's Historical Performance by Game.

```
arima_peytonManning <- arima(
  ts_peytonManning,
  order = c(1, 0, 1))

summary(arima_peytonManning)
```

```
Call:
arima(x = ts_peytonManning, order = c(1, 0, 1))

Coefficients:
         ar1      ma1  intercept
      0.9134  -0.8192    17.4587
s.e.  0.0605   0.0786     0.9682

sigma^2 estimated as 59.86:  log likelihood = -959.87,  aic = 1927.73

Training set error measures:
                      ME     RMSE      MAE       MPE     MAPE      MASE
Training set 0.005206747 7.736947 6.030506 -99.38311 124.2982 0.7309896
                     ACF1
Training set -0.008272646
```

```r
confint(arima_peytonManning)
```

```
                2.5 %     97.5 %
ar1         0.7949304  1.0319427
ma1        -0.9732963 -0.6650994
intercept 15.5611483 19.3563260
```

```r
forecast::checkresiduals(arima_peytonManning)
```

```
    Ljung-Box test

data:  Residuals from ARIMA(1,0,1) with non-zero mean
Q* = 3.8411, df = 8, p-value = 0.8712

Model df: 2.   Total lags used: 10
```

Figure 21.6 Model Summary of Autoregressive Integrated Moving Average Model fit to Peyton Manning's Historical Performance by Game.

21.3.6 Generate the Model Forecasts

```r
forecast_tomBrady <- forecast::forecast(
  arima_tomBrady,
  level = c(80, 95)) # 80% and 95% confidence intervals

forecast_peytonManning <- forecast::forecast(
  arima_peytonManning,
  level = c(80, 95)) # 80% and 95% confidence intervals

forecast_tomBrady
```

```
    Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
384       21.03918 11.290652 30.78772 6.1300916 35.94828
385       15.83752  6.050851 25.62418 0.8701029 30.80493
386       22.89837 13.062972 32.73377 7.8564254 37.94032
387       18.55187  8.527204 28.57654 3.2204660 33.88327
388       17.70611  7.680168 27.73206 2.3727526 33.03948
389       18.27131  8.148171 28.39445 2.7893058 33.75331
390       17.91113  7.751759 28.07050 2.3737135 33.44854
391       19.61626  9.454630 29.77790 4.0753848 35.15715
392       19.52851  9.362719 29.69431 3.9812732 35.07575
393       18.52801  8.361647 28.69436 2.9799013 34.07611
```

```r
forecast_peytonManning
```

```
    Point Forecast    Lo 80    Hi 80      Lo 95    Hi 95
278       12.71938 2.804079 22.63467 -2.4447620 27.88351
279       13.12963 3.170404 23.08886 -2.1016934 28.36096
280       13.50437 3.508639 23.50011 -1.7827835 28.79153
281       13.84668 3.820584 23.87277 -1.4869093 29.18027
282       14.15935 4.107997 24.21070 -1.2128686 29.53157
283       14.44496 4.372576 24.51734 -0.9594210 29.84933
284       14.70584 4.615948 24.79573 -0.7253186 30.13700
285       14.94414 4.839661 25.04862 -0.5093279 30.39761
286       15.16181 5.045178 25.27845 -0.3102454 30.63387
287       15.36064 5.233877 25.48741 -0.1269092 30.84819
```

21.3.7 Plot the Model Forecasts

```r
forecast::autoplot(forecast_tomBrady) +
  labs(
    x = "Game Number",
```

```
    y = "Fantasy Points",
    title = "Tom Brady's Historical and Projected Fantasy Points by Game",
    subtitle = "(if he were to have continued playing additional seasons)"
  ) +
  theme_classic()
```

Figure 21.7 Tom Brady's Historical and Projected Fantasy Points by Game.

```
forecast::autoplot(forecast_peytonManning) +
  labs(
    x = "Game Number",
    y = "Fantasy Points",
    title = "Peyton Manning's Historical and Projected Fantasy Points by Game",
    subtitle = "(if he were to have continued playing additional seasons)"
  ) +
  theme_classic()
```

Figure 21.8 Peyton Manning's Historical and Projected Fantasy Points by Game.

21.4 Bayesian Mixed Models

The Bayesian longitudinal mixed models were estimated in Section 12.3.5.

21.4.1 Prepare New Data Object

```
player_stats_seasonal_offense_subset <- player_stats_seasonal_offense %>%
  filter(position_group %in% c("QB","RB","WR","TE"))

player_stats_seasonal_offense_subset$position[which(player_stats_seasonal_offense_subset$position

player_stats_seasonal_kicking_subset <- player_stats_seasonal_kicking %>%
  filter(position == "K")

player_stats_seasonal_offense_subset <- bind_rows(
  player_stats_seasonal_offense_subset,
  player_stats_seasonal_kicking_subset
```

```
)

player_stats_seasonal_offense_subset$player_idFactor <- factor(player_stats_seasonal_offense_subse
player_stats_seasonal_offense_subset$positionFactor <- factor(player_stats_seasonal_offense_subset
```

```
player_stats_seasonal_offense_subsetCC <- player_stats_seasonal_offense_subset %>%
  filter(
    !is.na(player_idFactor),
    !is.na(fantasy_points),
    !is.na(positionFactor),
    !is.na(ageCentered20),
    !is.na(ageCentered20Quadratic),
    !is.na(years_of_experience))

player_stats_seasonal_offense_subsetCC <- player_stats_seasonal_offense_subsetCC %>%
  filter(player_id %in% bayesianMixedModelFit$data$player_idFactor) %>%
  mutate(positionFactor = droplevels(positionFactor))

player_stats_seasonal_offense_subsetCC <- player_stats_seasonal_offense_subsetCC %>%
  group_by(player_id) %>%
  group_modify(~ add_row(.x, season = max(player_stats_seasonal_offense_subsetCC$season) + 1)) %>%
  fill(player_display_name, player_idFactor, position, position_group, positionFactor, team, .dire
  ungroup

player_stats_seasonal_offense_subsetCC <- player_stats_seasonal_offense_subsetCC %>%
  left_join(
    player_stats_seasonal_offense_subsetCC %>%
      filter(season == max(player_stats_seasonal_offense_subsetCC$season) - 1) %>%
      select(player_id, age_lastYear = age, years_of_experience_lastYear = years_of_experience),
    by = "player_id") %>%
  mutate(
    age = if_else(season == max(player_stats_seasonal_offense_subsetCC$season), age_lastYear + 1,
    ageCentered20 = age - 20,
    years_of_experience = if_else(season == max(player_stats_seasonal_offense_subsetCC$season), ye

activePlayers <- unique(player_stats_seasonal_offense_subsetCC[c("player_id","season")]) %>%
  filter(season == max(player_stats_seasonal_offense_subsetCC$season) - 1) %>%
  select(player_id) %>%
  pull()

inactivePlayers <- player_stats_seasonal_offense_subsetCC$player_id[which(player_stats_seasonal_of

player_stats_seasonal_offense_subsetCC <- player_stats_seasonal_offense_subsetCC %>%
  filter(player_id %in% activePlayers | (player_id %in% inactivePlayers & season < max(player_stat
```

```
  mutate(
    player_idFactor = droplevels(player_idFactor)
  )
```

21.4.2 Generate Predictions

```
player_stats_seasonal_offense_subsetCC$fantasyPoints_bayesian <- predict(
  bayesianMixedModelFit,
  newdata = player_stats_seasonal_offense_subsetCC
)[,"Estimate"]
```

21.4.3 Table of Next Season Predictions

```
player_stats_seasonal_offense_subsetCC %>%
  filter(season == max(player_stats_seasonal_offense_subsetCC$season), position == "QB") %>%
  arrange(-fantasyPoints_bayesian) %>%
  select(player_display_name, fantasyPoints_bayesian)
```

```
# A tibble: 81 x 2
   player_display_name fantasyPoints_bayesian
   <chr>                                <dbl>
 1 Patrick Mahomes                       416.
 2 Josh Allen                            411.
 3 Lamar Jackson                         317.
 4 Jalen Hurts                           304.
 5 Justin Herbert                        297.
 6 Dak Prescott                          290.
 7 Jared Goff                            288.
 8 Aaron Rodgers                         267.
 9 Joe Burrow                            267.
10 Trevor Lawrence                       257.
# i 71 more rows
```

```
player_stats_seasonal_offense_subsetCC %>%
  filter(season == max(player_stats_seasonal_offense_subsetCC$season), position == "RB") %>%
  arrange(-fantasyPoints_bayesian) %>%
  select(player_display_name, fantasyPoints_bayesian)
```

```
# A tibble: 133 x 2
```

```
   player_display_name fantasyPoints_bayesian
   <chr>                                <dbl>
 1 Christian McCaffrey                   222.
 2 Jonathan Taylor                       190.
 3 Josh Jacobs                           181.
 4 Derrick Henry                         181.
 5 Travis Etienne                        177.
 6 Joe Mixon                             173.
 7 Najee Harris                          167.
 8 Alvin Kamara                          165.
 9 Jahmyr Gibbs                          161.
10 David Montgomery                      159.
# i 123 more rows
```

```
player_stats_seasonal_offense_subsetCC %>%
  filter(season == max(player_stats_seasonal_offense_subsetCC$season), position == "WR") %>%
  arrange(-fantasyPoints_bayesian) %>%
  select(player_display_name, fantasyPoints_bayesian)
```

```
# A tibble: 209 x 2
   player_display_name fantasyPoints_bayesian
   <chr>                                <dbl>
 1 CeeDee Lamb                           198.
 2 Justin Jefferson                      195.
 3 A.J. Brown                            192.
 4 Tyreek Hill                           191.
 5 Ja'Marr Chase                         186.
 6 DK Metcalf                            174.
 7 Amon-Ra St. Brown                     171.
 8 Deebo Samuel                          158.
 9 D.J. Moore                            157.
10 Tee Higgins                           151.
# i 199 more rows
```

```
player_stats_seasonal_offense_subsetCC %>%
  filter(season == max(player_stats_seasonal_offense_subsetCC$season), position == "TE") %>%
  arrange(-fantasyPoints_bayesian) %>%
  select(player_display_name, fantasyPoints_bayesian)
```

```
# A tibble: 112 x 2
   player_display_name fantasyPoints_bayesian
   <chr>                                <dbl>
 1 Travis Kelce                         181.
 2 Mark Andrews                         134.
```

```
 3 George Kittle                        131.
 4 T.J. Hockenson                       124.
 5 Sam LaPorta                          104.
 6 Cole Kmet                            100.
 7 Evan Engram                           99.0
 8 Dalton Schultz                        92.6
 9 Dallas Goedert                        91.9
10 Hunter Henry                          91.7
# i 102 more rows
```

21.4.4 Plot of Individuals' Model-Implied Predictions

21.4.4.1 Quarterbacks

```
plot_individualFantasyPointsByAgeQB <- ggplot(
  data = player_stats_seasonal_offense_subsetCC %>% filter(position == "QB"),
  mapping = aes(
    x = round(age, 2),
    y = round(fantasyPoints_bayesian, 2),
    group = player_id)) +
  geom_smooth(
    aes(
      x = age,
      y = fantasyPoints_bayesian,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    se = FALSE,
    linewidth = 0.5,
    color = "black") +
  geom_point(
    aes(
      x = age,
      y = fantasyPoints_bayesian,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    size = 1,
    color = "transparent" # make points invisible but keep tooltips
  ) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
```

```
    title = "Fantasy Points (Season) by Player Age: Quarterbacks"
  ) +
  theme_classic()

ggplotly(
  plot_individualFantasyPointsByAgeQB,
  tooltip = c("age","fantasyPoints_bayesian","text","label")
)
```

Figure 21.9 Plot of Individuals' Implied Trajectories of Fantasy Points by Age, from a Bayesian Generalized Additive Model, for Quarterbacks.

21.4.4.2 Running Backs

```
plot_individualFantasyPointsByAgeRB <- ggplot(
  data = player_stats_seasonal_offense_subsetCC %>% filter(position == "RB"),
  mapping = aes(
    x = age,
    y = fantasyPoints_bayesian,
    group = player_id)) +
  geom_smooth(
    aes(
      x = age,
      y = fantasyPoints_bayesian,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    se = FALSE,
    linewidth = 0.5,
    color = "black") +
  geom_point(
    aes(
      x = age,
      y = fantasyPoints_bayesian,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    size = 1,
    color = "transparent" # make points invisible but keep tooltips
  ) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age: Running Backs"
  ) +
  theme_classic()

ggplotly(
  plot_individualFantasyPointsByAgeRB,
  tooltip = c("age","fantasyPoints_bayesian","text","label")
)
```

Figure 21.10 Plot of Individuals' Implied Trajectories of Fantasy Points by Age, from a Bayesian Generalized Additive Model, for Running Backs.

21.4.4.3 Wide Receivers

```
plot_individualFantasyPointsByAgeWR <- ggplot(
  data = player_stats_seasonal_offense_subsetCC %>% filter(position == "WR"),
  mapping = aes(
    x = age,
    y = fantasyPoints_bayesian,
    group = player_id)) +
  geom_smooth(
    aes(
      x = age,
      y = fantasyPoints_bayesian,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    se = FALSE,
    linewidth = 0.5,
    color = "black") +
  geom_point(
    aes(
      x = age,
      y = fantasyPoints_bayesian,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    size = 1,
    color = "transparent" # make points invisible but keep tooltips
  ) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age: Wide Receivers"
  ) +
  theme_classic()

ggplotly(
  plot_individualFantasyPointsByAgeWR,
  tooltip = c("age","fantasyPoints_bayesian","text","label")
)
```

Figure 21.11 Plot of Individuals' Implied Trajectories of Fantasy Points by Age, from a Bayesian Generalized Additive Model, for Wide Receivers.

21.4.4.4 Tight Ends

```
plot_individualFantasyPointsByAgeTE <- ggplot(
  data = player_stats_seasonal_offense_subsetCC %>% filter(position == "TE"),
  mapping = aes(
    x = age,
    y = fantasyPoints_bayesian,
    group = player_id)) +
  geom_smooth(
    aes(
      x = age,
      y = fantasyPoints_bayesian,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    se = FALSE,
    linewidth = 0.5,
    color = "black") +
  geom_point(
    aes(
      x = age,
      y = fantasyPoints_bayesian,
      text = player_display_name, # add player name for mouse over tooltip
      label = season # add season for mouse over tooltip
    ),
    size = 1,
    color = "transparent" # make points invisible but keep tooltips
  ) +
  labs(
    x = "Player Age (years)",
    y = "Fantasy Points (Season)",
    title = "Fantasy Points (Season) by Player Age: Wide Receivers"
  ) +
  theme_classic()

ggplotly(
  plot_individualFantasyPointsByAgeTE,
  tooltip = c("age","fantasyPoints_bayesian","text","label")
)
```

Figure 21.12 Plot of Individuals' Implied Trajectories of Fantasy Points by Age, from a Bayesian Generalized Additive Model, for Tight Ends.

21.5 Conclusion

22

Decision Making in the Context of Uncertainty

22.1 Getting Started

22.1.1 Load Packages

```
library("petersenlab")
```

22.2 Overview of Decision Making in Uncertainty

22.3 Wisdom of the Crowd

In many domains, the average of forecasters' predictions is more accurate than the accuracy of the constituent individuals. In some domains, the average of non-expert forecasts is more accurate than the forecasts by individual experts. This phenomenon is called "collective intelligence", the "wisdom of the crowd", or the "wisdom of crowds" (Larrick et al., 2024; Rader et al., 2017; Simoiu et al., 2019; Surowiecki, 2005; Wagner & Vinaimont, 2010).

Aggregation of predictions from multiple people leverages several important features, including cognitive diversity and error cancellation.

Cognitive diversity refers to the representation of individuals with different perspectives because of their "differences in knowledge, training, experience, or thinking styles" (Rader et al., 2017, p. 8). Cognitive diversity is important because judgments from a cognitively homogeneous group will tend to err

systematically. That is, they tend to err in the same direction—either consistently above or below the truth; thus, their errors are correlated. By contrast, a cognitively diverse group will not tend to err systematically. Individuals of a cognitively diverse group will bring different areas of expertise (e.g., determination of player skill, player opportunity, matchup strength, etc.) to bear in making their predictions and will thus make different mistakes (Larrick et al., 2024). Consequently, for the people comprising a cognitively diverse group, their judgments will tend to err *randomly*—where some people's predictions fall above the truth and some people's predictions fall below the truth—i.e., individual judgments "bracket" the truth (Mannes et al., 2014). Thus, judgments from a cognitively diverse group tend to have uncorrelated errors.

Error cancellation deals with the idea that, when individuals' judgments bracket the truth and show random rather than systematic error, the average of the predictions will "cancel out" some of the errors so that the predictions average out to more closely approximate the truth. However, when individuals' judgments do not bracket the truth, the average of the predictions will not cancel out the errors.

Averaging projections from individuals tends to yield predictions that are more accurate than the accuracy of most forecasters in the group (Mannes et al., 2014). Indeed, when at least some of of the projections bracket the truth, averaged predictions must be more accurate than the average individual forecaster—in terms of mean absolute error (MAE)—and averaged predictions are often much more accurate (Larrick et al., 2024). When referring to the accuracy of the "average individual forecaster", we are referring to accuracy in terms of mean absolute error (MAE)—not to the accuracy of the forecaster at the 50th percentile. If none of the projections bracket the truth—e.g., all projections overestimate the truth—averaged predictions will be as accurate than the average individual forecaster in terms of mean absolute error (Larrick et al., 2024). In sum, "Averaging the answers of a crowd, therefore, ensures a level of accuracy no worse than the average member of the crowd and, in some cases, a level better than nearly all members" (Larrick et al., 2024, p. 126). Moreover, averaged projections tend to be more accurate than consensus-based judgments from groups of people that interact and discuss, due to cognitive biases associated with the social interaction among groups, such as herding in which people align their behavior with others (Mannes et al., 2014; Simoiu et al., 2019), though discussion can be helpful in some contexts (Larrick et al., 2024).

There are well-known prediction markets, in which people bet money to make predictions for various events, which allows determining the crowd-averaged prediction for events:

- https://www.predictit.org
- https://polymarket.com
- https://www.rotowire.com/betting/nfl/player-futures.php

Crowd-averaged projections tend to be most accurate when:

- the crowd consists of individuals who hold expertise in the domain such that they will make predictions that fall close to the truth
- there is relatively low variability in the expertise of the individual forecasters in terms of their ability to make accurate forecasts
- there is cognitive diversity among the forecasters
- the projections are made independently—i.e., the forecasters are not aware of others' forecasts and do not discuss or interact with the other forecasters
- the bracketing rate—i.e., the frequency with which any two forecasters' predictions fall on opposite sides of the truth—is high
- there are at least 5–10 sources of projections

However, the crowd is not more accurate than the expert or best forecaster in all situations or domains. For instance, the crowd tends to be less accurate than the (prospectively identified) best forecaster when there is great variability in forecasters' expertise (in terms of the forecasters' ability to forecast accurately) and when the bracketing rate is low (Mannes et al., 2014). Some forecasters may provide terrible projections; thus, including them in an average may make the average projections substantially less accurate. Thus, it may be necessary to examine the average of a "select crowd", by aggregating the projections of the most consistently accurate forecasters (Mannes et al., 2014). Incorporating at least 5–10 forecasters leverages most of the benefits of the crowd; adding additional forecasters tends to result in diminishing returns (Larrick et al., 2024). However, to the extent that those who are most accurate in a given period reflects luck, you are better off averaging the predictions of all forecasters than selecting the forecasters who were most accurate in the most recent period (Larrick et al., 2024).

22.4 Accuracy of Fantasy Football Crowd Projections

Even though the crowd tends to be more accurate than individual forecasters, crowd-averaged projections (at least among experts) are not necessarily highly accurate. In fantasy football, [DESCRIBE THEIR ACCURACY–BOTH FOR ALL PLAYERS AND WHEN SUBSETTING TO THE TOP X PLAYERS]. Nevertheless, individual sources tend to be even less accurate. [DESCRIBE THEIR ACCURACY].

The `petersenlab` package[1] (Petersen, 2024a) has the `wisdomOfCrowd()` function that computes the overall accuracy of the crowd-averaged projections,

[1] https://cran.r-project.org/web/packages/petersenlab/index.html

including the bracketing rate of the individual projections.

```
# insert example using historical projections: wisdomOfCrowd()
```

```
predictedValues <- c(10,20,30,40,60,70,80,90)
actualValue <- 50
errors <- predictedValues - actualValue

errors
```

```
[1] -40 -30 -20 -10  10  20  30  40
```

```
median(errors)
```

```
[1] 0
```

```
mean(predictedValues)
```

```
[1] 50
```

```
petersenlab::wisdomOfCrowd(
  predicted = predictedValues,
  actual = actualValue
)
```

```
              ME MAE MSE     RMSE MPE MAPE    sMAPE MASE     RMSLE
individual     0  25 750 27.38613   0   50 27.88011  Inf 0.7182053
crowdAveraged  0   0   0  0.00000   0    0  0.00000  NaN 0.0000000
              bracketingRate
individual         0.5714286
crowdAveraged      0.5714286
```

```
predictedValues <- c(10,29,29,40,60,70,80,90)
actualValue <- 71
errors <- predictedValues - actualValue

errors
```

```
[1] -61 -42 -42 -31 -11  -1   9  19
```

```r
median(errors)
```

```
[1] -21
```

```r
mean(predictedValues)
```

```
[1] 51
```

```r
petersenlab::wisdomOfCrowd(
  predicted = predictedValues,
  actual = actualValue
)
```

```
               ME MAE     MSE     RMSE      MPE     MAPE    sMAPE MASE
individual    -20  27 1096.75 33.11722 28.16901 38.02817 26.76303  Inf
crowdAveraged -20  20  400.00 20.00000 28.16901 28.16901 16.39344  Inf
                  RMSLE bracketingRate
individual    0.8273563      0.4285714
crowdAveraged 0.3254224      0.4285714
```

```r
predictedValues <- c(10,20,30,40,60,70,80,90)
actualValue <- 100
errors <- predictedValues - actualValue

errors
```

```
[1] -90 -80 -70 -60 -40 -30 -20 -10
```

```r
median(errors)
```

```
[1] -50
```

```r
mean(predictedValues)
```

```
[1] 50
```

```r
petersenlab::wisdomOfCrowd(
  predicted = predictedValues,
  actual = actualValue
)
```

```
               ME MAE  MSE     RMSE MPE MAPE    sMAPE MASE     RMSLE
individual    -50  50 3250 57.00877  50   50 38.02618  Inf 1.1196415
crowdAveraged -50  50 2500 50.00000  50   50 33.33333  Inf 0.6832949
              bracketingRate
individual                 0
crowdAveraged              0
```

Even though some sources are more accurate than the average in a given year, they are not *consistently* more accurate than the average. Prediction involves a combination of luck and skill. In some years, a prediction will invariably do better than others, in part, based on luck. However, luck is unlikely to continue systematically into future years, so a source that got lucky in a given year is likely to regress to the mean in subsequent years. That is, determinining the most accurate source in a given year, after the fact, is not necessarily the same as identifying the most skilled forecaster. It is easy to identify the most accurate source after the fact, but it is challenging to predict, in advance, who the best forecaster will be (Larrick et al., 2024). It requires a large sample of predictions to determine whether a given forecaster is reliably (i.e., consistently) more accurate than other forecasters and to identify the most accurate forecaster (Larrick et al., 2024). Thus, it can be challenging to know, in advance, who the most accurate forecasters will be. Because average projections are as or more accurate than the average forecaster's prediction, averaging projections across all forecasters is superior to choosing individual forecasters when the forecasters are roughly similar in forecasting ability or when it is hard to distinguish their ability in advance (Larrick et al., 2024).

The relatively modest accuracy of the projections by so-called fantasy "experts"' and of their average of their projections could occur for a number of reasons. One possibility is that the level of expertise of the "expert" forecasters in terms of being able to provide accurate forecasts is not strong. That is, because football performance and injuries are so challenging to predict, individual forecasters' projections may not be particularly close to the truth.

A second possibility is that the bracketing rate of the predictions is not particularly high (Mannes et al., 2014). Even if the individual forecasters' projections are not close to the truth, if ~50% of them overestimate the truth and the other 50% of the underestimate the truth, the average will more closely approximate the truth. However, if all forecasters overestimate the truth for a given player, averaging the projections will not necessarily lead to more accurate projections.

A third possibility is that the forecasts of the different experts are not independent.

Each of these possibilities is likely true to some degree. First, individuals' predictions are unlikely to be highly accurate consistently. Second, there are many players who are systematically *over*predicted (e.g., due to their injury)

or *under*predicted (e.g., due to their becoming the starter after a teammate becomes injured, is traded, etc.)—an example of overextremity miscalibration. In general, it is likely for players who are projected to score more points to be overpredicted and for players who are projected to score fewer points to be underpredicted [PRESENT CALIBRATION STATS]. Third, the experts may interact and discuss with one another. Interaction and discussion among experts may lead them to follow the herd and conform their projections to what each other predict. This has been terms informational influence and may reflect the anchoring and adjustment heuristic (Larrick et al., 2024). In any case, they are able to see each other's projections and make change their projections, accordingly.

22.5 How Well Do People Incorporate Advice From Others?

An important question is how well people incorporate advice from others. In the context of fantasy football, advice might be sources of projections. In general, evidence from social psychology suggests that people tend to underweight how much weight they give to advice from others relative to their own opinions, a phenomeon called egocentric discounting (Larrick et al., 2024; Rader et al., 2017). People tend to weight others' advice around 30% in terms of the proportion of a shift a person makes toward another person's perspective, though this depends on the perceived accuracy of the advisor. Moreover, people frequently ignores others' advice entirely. In general, people make use of crowds too little—they put too much weight on their own prediction and not enough weight on others' predictions whose diversity can be leveraged for error cancellation (Larrick et al., 2024).

22.6 Suggestions

Based on the above dicussion, here are some suggestions for decision making in the context of uncertainty:

- seek advice from diverse perspectives and incorporate it into your decision making
- try to get opinions from others *before* you state your perspective and before the various sources of advice discuss with each other, to ensure independence

of advice

  - As noted by Kahneman (2011), "before an issue is discussed, all members of the committee should be asked to write a very brief summary of their position… The standard practice of open discussion gives too much weight to the opinions of those who speak early and assertively, causing others to line up behind them." (p. 85).

- if some sources of advice (or projections) are clearly more skilled and accurate than others, you can average this "select" crowd of projections or give them greater weight in a weighted average
- if it is unclear whether or which sources are reliably more accurate than others, using a simple average across all sources (i.e., crowd-averaged projections) can be a useful approach that is as accurate as—if not more accurate than—the average individual forecaster
- incorporate at least 5–10 sources of projections

22.7 Conclusion

References

Ægisdóttir, S., White, M. J., Spengler, P. M., Maugherman, A. S., Anderson, L. A., Cook, R. S., Nichols, C. N., Lampropoulos, G. K., Walker, B. S., Cohen, G., & Rush, J. D. (2006). The meta-analysis of clinical judgment project: Fifty-six years of accumulated research on clinical versus statistical prediction. *The Counseling Psychologist*, *34*(3), 341–382. https://doi.org/10.1177/0011000005285875

Akinshin, A. (2023). Weighted quantile estimators. *arXiv*. https://doi.org/10.48550/arXiv.2304.07265

Andersen, D., Petersen, I. T., & Tungate, A. (2024). *ffanalytics: Scrape data for fantasy football*. https://github.com/FantasyFootballAnalytics/ffanalytics

Atanasov, P., Witkowski, J., Ungar, L., Mellers, B., & Tetlock, P. (2020). Small steps to accuracy: Incremental belief updaters are better forecasters. *Organizational Behavior and Human Decision Processes*, *160*, 19–35. https://doi.org/10.1016/j.obhdp.2020.02.001

Austin, P. C., & Steyerberg, E. W. (2014). Graphical assessment of internal and external calibration of logistic regression models by using loess smoothers. *Statistics in Medicine*, *33*(3), 517–535. https://doi.org/10.1002/sim.5941

Avugos, S., Köppen, J., Czienskowski, U., Raab, M., & Bar-Eli, M. (2013). The "hot hand" reconsidered: A meta-analytic approach. *Psychology of Sport and Exercise*, *14*(1), 21–27. https://doi.org/10.1016/j.psychsport.2012.07.005

Baird, C., & Wagner, D. (2000). The relative validity of actuarial- and consensus-based risk assessment systems. *Children and Youth Services Review*, *22*(11), 839–871. https://doi.org/10.1016/S0190-7409(00)00122-5

Bar-Eli, M., Avugos, S., & Raab, M. (2006). Twenty years of "hot hand" research: Review and critique. *Psychology of Sport and Exercise*, *7*(6), 525–553. https://doi.org/10.1016/j.psychsport.2006.03.001

Bocskocsky, A., Ezekowitz, J., & Stein, C. (2014). The hot hand: A new approach to an old "fallacy." *MIT Sloan Sports Analytics Conference*.

Bolger, F., & Önkal-Atay, D. (2004). The effects of feedback on judgmental interval predictions. *International Journal of Forecasting*, *20*(1), 29–39. https://doi.org/10.1016/S0169-2070(03)00009-8

Chang, W. (2018). *R graphics cookbook: Practical recipes for visualizing data* (2nd ed.). O'Reilly Media. https://r-graphics.org

Chatterjee, S. (2021). A new coefficient of correlation. *Journal of the American Statistical Association*, *116*(536), 2009–2022. https://doi.org/10.1080/01621459.2020.1758115

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Lawrence Erlbaum Associates, Publishers. https://doi.org/10.4324/9780203771587

Congelio, B. J. (2023). *Introduction to NFL analytics with R*. CRC Press. https://bradcongelio.com/nfl-analytics-with-r-book

Corston, R., & Colman, A. M. (2000). *A crash course in SPSS for Windows*. Wiley-Blackwell.

Critcher, C. R., & Rosenzweig, E. L. (2014). The performance heuristic: A misguided reliance on past success when predicting prospects for improvement. *Journal of Experimental Psychology: General*, *143*(2), 480–485. https://doi.org/10.1037/a0034129

D'Onofrio, B. M., Sjölander, A., Lahey, B. B., Lichtenstein, P., & Öberg, A. S. (2020). Accounting for confounding in observational studies. *Annual Review of Clinical Psychology*, *16*(1), 25–48. https://doi.org/10.1146/annurev-clinpsy-032816-045030

Dana, J., & Thomas, R. (2006). In defense of clinical judgment … and mechanical prediction. *Journal of Behavioral Decision Making*, *19*(5), 413–428. https://doi.org/10.1002/bdm.537

Dawes, R. M., Faust, D., & Meehl, P. E. (1989). Clinical versus actuarial judgment. *Science*, *243*(4899), 1668–1674. https://doi.org/10.1126/science.2648573

Den Hartigh, R. J. R., Niessen, A. S. M., Frencken, W. G. P., & Meijer, R. R. (2018). Selection procedures in sports: Improving predictions of athletes' future performance. *European Journal of Sport Science*, *18*(9), 1191–1198. https://doi.org/10.1080/17461391.2018.1480662

Digitale, J. C., Martin, J. N., & Glymour, M. M. (2022). Tutorial on directed acyclic graphs. *Journal of Clinical Epidemiology*, *142*, 264–267. https://doi.org/10.1016/j.jclinepi.2021.08.001

Eddy, D. M. (1982). Probabilistic reasoning in clinical medicine: Problems and opportunities. In D. Kahneman, P. Slovic, & A. Tversky (Eds.), *Judgment under uncertainty: Heuristics and biases* (pp. 249–267). Cambridge University Press.

Enke, B. (2020). What you see is all there is. *The Quarterly Journal of Economics*, *135*(3), 1363–1398. https://doi.org/10.1093/qje/qjaa012

Farrington, D. P., & Loeber, R. (1989). Relative improvement over chance (RIOC) and phi as measures of predictive efficiency and strength of association in 2×2 tables. *Journal of Quantitative Criminology*, *5*(3), 201–213. https://doi.org/10.1007/BF01062737

Gandrud, C. (2020). *Reproducible research with R and R studio* (3rd ed.). CRC Press. https://www.routledge.com/Reproducible-Research-with-R-and-RStudio/Gandrud/p/book/9780367143985

Garb, H. N., & Wood, J. M. (2019). Methodological advances in statistical

prediction. *Psychological Assessment*, *31*(12), 1456–1466. https://doi.org/10.1037/pas0000673

Getty, D., Li, H., Yano, M., Gao, C., & Hosoi, A. E. (2018). Luck and the law: Quantifying chance in fantasy sports and other contests. *SIAM Review*, *60*(4), 869–887. https://doi.org/10.1137/16m1102094

Gilovich, T., Vallone, R., & Tversky, A. (1985). The hot hand in basketball: On the misperception of random sequences. *Cognitive Psychology*, *17*(3), 295–314. https://doi.org/10.1016/0010-0285(85)90010-6

Goodman, S. (2008). A dirty dozen: Twelve *p*-value misconceptions. *Seminars in Hematology*, *45*(3), 135–140. https://doi.org/10.1053/j.seminhematol.2008.04.003

Grove, W. M., & Meehl, P. E. (1996). Comparative efficiency of informal (subjective, impressionistic) and formal (mechanical, algorithmic) prediction procedures: The clinical–statistical controversy. *Psychology, Public Policy, and Law*, *2*(2), 293–323. https://doi.org/10.1037/1076-8971.2.2.293

Grove, W. M., Zald, D. H., Lebow, B. S., Snitz, B. E., & Nelson, C. (2000). Clinical versus mechanical prediction: A meta-analysis. *Psychological Assessment*, *12*(1), 19–30. https://doi.org/10.1037/1040-3590.12.1.19

Harrell, Jr., F. E. (2024). *rms: Regression modeling strategies*. https://hbiostat.org/R/rms/

Hoch, S. J. (1985). Counterfactual reasoning and accuracy in predicting personal events. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *11*(4), 719–731. https://doi.org/10.1037/0278-7393.11.1-4.719

Hough, S. E. (2016). *Predicting the unpredictable: The tumultuous science of earthquake prediction*. Princeton University Press.

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and practice* (3rd ed.). OTexts. https://otexts.com/fpp3

Iacobucci, D., Schneider, M. J., Popovich, D. L., & Bakamitsos, G. A. (2016). Mean centering helps alleviate "micro" but not "macro" multicollinearity. *Behavior Research Methods*, *48*(4), 1308–1317. https://doi.org/10.3758/s13428-015-0624-x

Johnson, J. E. V., & Bruce, A. C. (2001). Calibration of subjective probability judgments in a naturalistic setting. *Organizational Behavior and Human Decision Processes*, *85*(2), 265–290. https://doi.org/10.1006/obhd.2000.2949

Kahneman, D. (2011). *Thinking, fast and slow*. Farrar, Straus, and Giroux.

Kassambara, A. (2017). *Practical guide to cluster analysis in R: Unsupervised machine learning* (Vol. 1). Sthda.

Keren, G. (1987). Facing uncertainty in the game of bridge: A calibration study. *Organizational Behavior and Human Decision Processes*, *39*(1), 98–114. https://doi.org/10.1016/0749-5978(87)90047-1

Kessler, R. C., Bossarte, R. M., Luedtke, A., Zaslavsky, A. M., & Zubizarreta, J. R. (2020). Suicide prediction models: A critical review of recent research with recommendations for the way forward. *Molecular Psychiatry*, *25*(1), 168–179. https://doi.org/10.1038/s41380-019-0531-0

Kievit, R., Frankenhuis, W., Waldorp, L., & Borsboom, D. (2013). Simpson's paradox in psychological science: A practical guide. *Frontiers in Psychology*, *4*(513). https://doi.org/10.3389/fpsyg.2013.00513

Koehler, D. J., Brenner, L., & Griffin, D. (2002). The calibration of expert judgment: Heuristics and biases beyond the laboratory. In T. Gilovich, D. Griffin, & D. Kahneman (Eds.), *Heuristics and biases: The psychology of intuitive judgment*. Cambridge University Press.

Koriat, A., Lichtenstein, S., & Fischhoff, B. (1980). Reasons for confidence. *Journal of Experimental Psychology: Human Learning and Memory*, *6*(2), 107–118. https://doi.org/10.1037/0278-7393.6.2.107

Kotrba, V. (2020). Heuristics in fantasy sports: Is it profitable to strategize based on favourite of the match? *Mind & Society*, *19*(1), 195–206. https://doi.org/10.1007/s11299-020-00231-7

Larrick, R. P., Mannes, A. E., & Soll, J. B. (2024). The social psychology of the wisdom of crowds (with a new section on recent advances). In F. M. Federspiel, G. Montibeller, & M. Seifert (Eds.), *Behavioral decision analysis* (pp. 121–143). Springer. https://doi.org/10.1007/978-3-031-44424-1_7

Lederer, D. J., Bell, S. C., Branson, R. D., Chalmers, J. D., Marshall, R., Maslove, D. M., Ost, D. E., Punjabi, N. M., Schatz, M., Smyth, A. R., Stewart, P. W., Suissa, S., Adjei, A. A., Akdis, C. A., Azoulay, É., Bakker, J., Ballas, Z. K., Bardin, P. G., Barreiro, E., … Vincent, J.-L. (2019). Control of confounding and reporting of results in causal inference studies. Guidance for authors from editors of respiratory, sleep, and critical care journals. *Annals of the American Thoracic Society*, *16*(1), 22–28. https://doi.org/10.1513/AnnalsATS.201808-564PS

Lee, M. D., & Liu, S. (2022). Drafting strategies in fantasy football: A study of competitive sequential human decision making. *Judgment and Decision Making*, *17*(4), 691–719. https://doi.org/10.1017/S1930297500008901

Lilienfeld, S. O. (2007). Psychological treatments that cause harm. *Perspectives on Psychological Science*, *2*(1), 53–70. https://doi.org/10.1111/j.1745-6916.2007.00029.x

Lindhiem, O., Petersen, I. T., Mentch, L. K., & Youngstrom, E. A. (2020). The importance of calibration in clinical psychology. *Assessment*, *27*(4), 840–854. https://doi.org/10.1177/1073191117752055

Lyons, B. D., Hoffman, B. J., Michel, J. W., & Williams, K. J. (2011). On the predictive efficiency of past performance and physical ability: The case of the national football league. *Human Performance*, *24*(2), 158–172. https://doi.org/10.1080/08959285.2011.555218

Makridakis, S., Hogarth, R. M., & Gaba, A. (2009). Forecasting and uncertainty in the economic and business world. *International Journal of Forecasting*, *25*(4), 794–812. https://doi.org/10.1016/j.ijforecast.2009.05.012

Mannes, A. E., Soll, J. B., & Larrick, R. P. (2014). The wisdom of select crowds. *Journal of Personality and Social Psychology*, *107*(2), 276–299. https://doi.org/10.1037/a0036677

McGrath, R. E., & Meyer, G. J. (2006). When effect sizes disagree: The case of

r and *d*. *Psychological Methods*, *11*(4), 386–401. https://doi.org/10.1037/1082-989X.11.4.386

Meehl, P. E. (1957). When shall we use our heads instead of the formula? *Journal of Counseling Psychology*, *4*(4), 268–273. https://doi.org/10.1037/h0047554

Meehl, P. E. (1978). Theoretical risks and tabular asterisks: Sir Karl, Sir Ronald, and the slow progress of soft psychology. *Journal of Consulting and Clinical Psychology*, *46*(4), 806–834. https://doi.org/10.1037/0022-006x.46.4.806

Meehl, P. E. (1986). Causes and effects of my disturbing little book. *Journal of Personality Assessment*, *50*(3), 370–375. https://doi.org/10.1207/s15327752jpa5003_6

Meehl, P. E., & Rosen, A. (1955). Antecedent probability and the efficiency of psychometric signs, patterns, or cutting scores. *Psychological Bulletin*, *52*(3), 194–216. https://doi.org/10.1037/h0048070

Miller, J. B., & Sanjurjo, A. (2014). A cold shower for the hot hand fallacy. *Innocenzo Gasparini Institute for Economic Research*. https://repec.unibocconi.it/igier/igi/wp/2014/518.pdf

Miller, R. M. (2013). *Cognitive bias in fantasy sports: Is your brain sabotaging your team?* Xlibris Press.

Mlodinow, L. (2008). *The drunkard's walk: How randomness rules our lives*. Pantheon Books.

Moore, D. A., & Healy, P. J. (2008). The trouble with overconfidence. *Psychological Review*, *115*(2), 502–517. https://doi.org/10.1037/0033-295X.115.2.502

Morley, S. K., Brito, T. V., & Welling, D. T. (2018). Measures of model performance based on the log accuracy ratio. *Space Weather*, *16*(1), 69–88. https://doi.org/10.1002/2017SW001669

Motz, B. (2013). Fantasy football: A touchdown for undergraduate statistics education. *Proceedings of the Games, Learning, and Society Conference*, *9.0*, 222–228. https://doi.org/10.1184/R1/6686804.v1

Murphy, A. H., & Winkler, R. L. (1984). Probability forecasting in meterology. *Journal of the American Statistical Association*, *79*(387), 489–500. https://doi.org/10.2307/2288395

Oskamp, S. (1965). Overconfidence in case-study judgments. *Journal of Consulting Psychology*, *29*(3), 261–265. https://doi.org/10.1037/h0022125

Pelechrinis, K., & Winston, W. (2022). The hot hand in the wild. *PLOS ONE*, *17*(1), e0261890. https://doi.org/10.1371/journal.pone.0261890

Petersen, I. T. (2024a). *petersenlab: A collection of R functions by the Petersen Lab*. https://github.com/DevPsyLab/petersenlab

Petersen, I. T. (2024c). *Principles of psychological assessment: With applied examples in R*. University of Iowa Libraries. https://doi.org/10.25820/work.007199

Petersen, I. T. (2024b). *Principles of psychological assessment: With applied examples in R*. Chapman and Hall/CRC. https://doi.org/10.1201/

9781003357421

Rader, C. A., Larrick, R. P., & Soll, J. B. (2017). Advice as a form of social influence: Informational motives and the consequences for accuracy. *Social and Personality Psychology Compass*, *11*(8), e12329. https://doi.org/https://doi.org/10.1111/spc3.12329

Rice, M. E., Harris, G. T., & Lang, C. (2013). Validation of and revision to the VRAG and SORAG: The Violence Risk Appraisal Guide—Revised (VRAG-R). *Psychological Assessment*, *25*(3), 951–965. https://doi.org/10.1037/a0032878

Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J.-C., & Müller, M. (2023). *pROC: Display and analyze ROC curves*. https://xrobin.github.io/pROC/

Rohrer, J. M. (2018). Thinking clearly about correlations and causation: Graphical causal models for observational data. *Advances in Methods and Practices in Psychological Science*, *1*(1), 27–42. https://doi.org/10.1177/2515245917745629

Russo, J. E., & Schoemaker, P. J. (1992). Managing overconfidence. *Sloan Management Review*, *33*(2), 7.

Schwabish, J. (2021). *Better data visualizations: A guide for scholars, researchers, and wonks*. Columbia University Press.

Scrucca, L., Fraley, C., Murphy, T. B., & Raftery, A. E. (2023). *Model-based clustering, classification, and density estimation using mclust in R*. Chapman; Hall/CRC.

Shadish, W. R., Cook, T. D., & Campbell, D. T. (2002). *Experimental and quasi-experimental designs for generalized causal inference*. Houghton Mifflin.

Silver, N. (2012). *The signal and the noise: Why so many predictions fail–but some don't*. Penguin.

Simoiu, C., Sumanth, C., Mysore, A., & Goel, S. (2019). Studying the "wisdom of crowds" at scale. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, *7*(1), 171–179. https://doi.org/10.1609/hcomp.v7i1.5271

Skala, D. (2008). Overconfidence in psychology and finance–an interdisciplinary literature review. *Bank i Kredyt*, *4*, 33–50.

Smith, B., Sharma, P., & Hooper, P. (2006). Decision making in online fantasy sports communities. *Interactive Technology and Smart Education*, *3*(4), 347–360. https://doi.org/10.1108/17415650680000072

Spector, P. E., & Brannick, M. T. (2010). Methodological urban legends: The misuse of statistical control variables. *Organizational Research Methods*, *14*(2), 287–305. https://doi.org/10.1177/1094428110369842

Stevens, R. J., & Poppe, K. K. (2020). Validation of clinical prediction models: What does the "calibration slope" really measure? *Journal of Clinical Epidemiology*, *118*, 93–99. https://doi.org/10.1016/j.jclinepi.2019.09.016

Steyerberg, E. W., & Vergouwe, Y. (2014). Towards better clinical prediction models: Seven steps for development and an ABCD for vali-

dation. *European Heart Journal*, *35*(29), 1925–1931. https://doi.org/10.1093/eurheartj/ehu207

Surowiecki, J. (2005). *The wisdom of crowds*. Anchor Books.

Tetlock, P. E. (2017). *Expert political judgment: How good is it? How can we know? - New edition*. Princeton University Press.

Textor, J., Zander, B. van der, Gilthorpe, M. S., Liśkiewicz, M., & Ellison, G. T. (2017). Robust causal inference using directed acyclic graphs: The R package "dagitty". *International Journal of Epidemiology*, *45*(6), 1887–1894. https://doi.org/10.1093/ije/dyw341

Tofallis, C. (2015). A better measure of relative prediction accuracy for model selection and model estimation. *Journal of the Operational Research Society*, *66*(8), 1352–1362. https://doi.org/10.1057/jors.2014.103

Treat, T. A., & Viken, R. J. (2023). Measuring test performance with signal detection theory techniques. In H. Cooper, M. N. Coutanche, L. M. McMullen, A. T. Panter, D. Rindskopf, & K. J. Sher (Eds.), *APA handbook of research methods in psychology: Foundations, planning, measures, and psychometrics* (2nd ed., Vol. 1, pp. 837–858). American Psychological Association.

Tufte, E. R. (2001). *The visual display of quantitative information*. Graphics Press.

Tversky, A., & Kahneman, D. (1974). Judgment under uncertainty: Heuristics and biases. *Science*, *185*(4157), 1124–1131. https://doi.org/10.1126/science.185.4157.1124

Ursenbach, J., O'Connell, M. E., Neiser, J., Tierney, M. C., Morgan, D., Kosteniuk, J., & Spiteri, R. J. (2019). Scoring algorithms for a computer-based cognitive screening tool: An illustrative example of overfitting machine learning approaches and the impact on estimates of classification accuracy. *Psychological Assessment*, *31*(11), 1377–1382. https://doi.org/10.1037/pas0000764

Wagner, C., & Vinaimont, T. (2010). Evaluating the wisdom of crowds. *Issues in Information Systems*, *11*(1), 724–732. http://iacis.org/iis/2010/724-732_LV2010_1546.pdf

White, M. H., & Sheldon, K. M. (2014). The contract year syndrome in the NBA and MLB: A classic undermining pattern. *Motivation and Emotion*, *38*(2), 196–205. https://doi.org/10.1007/s11031-013-9389-7

Wickham, H. (2024). *ggplot2: Elegant graphics for data analysis* (3rd ed.). Springer. https://ggplot2-book.org

Williams, A. J., Botanov, Y., Kilshaw, R. E., Wong, R. E., & Sakaluk, J. K. (2021). Potentially harmful therapies: A meta-scientific review of evidential value. *Clinical Psychology: Science and Practice*, *28*(1), 5–18. https://doi.org/10.1111/cpsp.12331

Woodland, L. M., & Woodland, B. M. (2015). The National Football League season wins total betting market: The impact of heuristics on behavior. *Southern Economic Journal*, *82*(1), 38–54. https://doi.org/10.4284/0038-4038-2013.145

Wysocki, A. C., Lawson, K. M., & Rhemtulla, M. (2022). Statistical control requires causal justification. *Advances in Methods and Practices in Psychological Science*, *5*(2), 25152459221095823. https://doi.org/10.1177/25152459221095823

Index

correlation, 323, 324

GitHub, xix

positive likelihood ratio, 564