Advanced Epidemiological Methods

M. Ehsan. Karim Epi-OER team

2024-04-20

Table of contents

The Project **14**
What We Aim to Achieve 14
Dive into Our Modules 15
How Our Content is Presented 17
Open Copyright License 17
Grant Applicants 17
Acknowledgements 18
How to Cite 18

I Data wrangling **20**
Background 21
Overview of tutorials 22

R basics **25**

Data types **35**

Automating tasks **42**

Importing dataset **47**

Data manipulation **51**

Import external data **60**

Summary tables **68**

R Markdown **73**
Introduction 73
Prerequisites 74
Knitting RMD 74
Tips and Troubleshooting 80

R Functions (W) **83**

Quiz (W) **85**
Live Quiz 85
Download Quiz 85

App (W) **87**

Exercise (W) **88**
Problem Statement 89
Problem 1: Basic Manipulation [60%] 89
Problem 2: Table 1 [10%] 91
Problem 3: Table 1 for subset [10%] 92
Problem 4: Considering eligibility criteria [20%] 92
Optional [0%] 92

II Accessing data **94**
Background 95
Overview of tutorials 95

Concepts (A) **98**
Model-based approach 98
Design-based approach 98
Reading list 98
Video Lessons 99
Video Lesson Slides 99
Links 99
References 100

Survey data sources **101**

Descriptions **103**

Importing CCHS to R **111**

Importing NHANES to R **135**

Reproducing NHANES results **142**
References 152

Importing NHIS to R **153**

Linking mortality data **163**

R Functions (A) **172**

Quiz (A) **173**
Live Quiz 173
Download Quiz 173

App (A) **175**

Exercise (A) **176**
Problem Statement 176
Question 1: [60% grade] 177
Question 2: Table 1 [20% grade] 179
Question 3: [20% grade] 181

III Research questions **183**
Background 184
Overview of tutorials 184

Concepts (Q) **186**
PICOT Framework 186
FINER Criteria 186
SAP 187
Video Lessons 188
Video Lesson Slides 189
Links 189
References 189

Predictive question-1 **190**

Predictive question-2a **200**
Saving data for later use 214
Exercise (try yourself) 214
References 214

Predictive question-2b **215**
Saving for further use 223
Regression summary (Optional) 223
Check missingness (optional) 227
Exercise (try yourself) 236
References 236

Causal question-1 **238**
Naive Analysis of combined 3 cycles 259
Save data for later 267
References 267

Causal question-2 **268**

R functions (Q) **286**

Quiz (Q) **287**
Live Quiz 287
Download Quiz 287

App (Q) **289**

IV Causal roles **290**
Background 291
Overview of tutorials 291

Concepts (R) **294**
Confounding 294
Effect modifier 295
Table 2 fallacy 295
Reading list 296
Video Lessons 296
Video Lesson Slides 297
Links 298
References 298

Confounding **299**

Mediator **305**

Collider **311**

Z-bias **316**

Collapsibility **321**

Change-in-estimate **338**
Adjusting for a variable that is a confounder 338

Adjusting for a variable that is not a confounder (simplified) . . . 342
Adjusting for a variable that is not a confounder (Complex) . . . 347

R functions (R) **352**

Quiz (R) **353**
Live Quiz . . . 353
Download Quiz . . . 353
Quiz (R) . . . 353

V Prediction ideas **355**
Background . . . 356
Overview of tutorials . . . 356

Concepts (P) **360**
Research goals . . . 360
Reading list . . . 360
Video Lessons . . . 361
Video Lesson Slides . . . 362
Links . . . 362
References . . . 362

Collinear predictors **363**

Continuous outcome **376**

Binary outcome **399**

Overfitting and performance **407**

Data Splitting **418**

Cross-validation **424**

Bootstrap **430**

R functions (P) **447**

Quiz (P) **449**
Live Quiz . . . 449

Download Quiz 449

VI Survey data analysis **451**
Background 452
Overview of tutorials 452

Concepts (D) **457**
Survey Data Analysis 457
Reading list 457
Video Lessons 458
Video Lesson Slides 459
Links 459
References 459

CCHS: Revisiting PICOT **460**
References 484

CCHS: Assessing data **485**

CCHS: Bivariate analysis **504**

CCHS: Regression **514**

CCHS: Performance **530**

NHANES: Blood Pressure **539**
A note about Predictive models 549

NHANES: Cholesterol **551**

NHANES: Subsetting **570**

R functions (D) **576**

Quiz (D) **577**
Live Quiz 577
Download Quiz 577

App (D) **579**

Exercise (D) **580**
Problem Statement 580

Question 1: Creating data and table 581
Question 2 584

VII Missing data analysis **588**
Background 589
Overview of tutorials 589

Concepts (M) **594**
Missing Data Analysis 594
Reading list 595
Video Lessons 595
Video Lesson Slides 595
Links 596
References 596

Imputation **597**

Imputation in NHANES **644**

Missing in outcome **668**

Performance with NA **683**

Subpopulations **700**

MCAR tests **717**

Effect modification **735**

R functions (M) **740**

Quiz (M) **741**
Live Quiz 741
Download Quiz 741

App (M) **743**

Exercise (M) **744**
Problem Statement 744
Question 1: Analytic dataset 746
Question 2: Dealing with missing values in confoud-
ners [100% grade] 749

Question 3: Dealing with missing values in outcome, predictor, and confoudners [optional] 751

VIIIPropensity score **753**
Background 754
Overview of tutorials 755
Propensity score weighting 757
Propensity score matching with multiple imputation in subpopulations 758
Propensity score weighting with multiple imputation in subpopulations 758
Propensity score weighting for multiple treatment categories 758

Concepts (S) **760**
Propensity Score Analysis 760
Reading list 760
Video Lessons 761
Video Lesson Slides 762
Links 762
References 763

Exact Matching (CCHS) **764**

PSM in OA-CVD (CCHS) **782**

PSM in OA-CVD (US) **818**

PSM in BMI-diabetes **841**
References 855

PSM with MI **857**

PS Weighting (US) **874**

PSM with MI in subset **897**

PSW with MI in subset **919**

PSW with multiple tx **935**

R functions (S) **962**

Quiz (S) **963**
Live Quiz 963
Download Quiz 963

App (S) **965**

Exercise (S) **966**
Problem Statement 966
Question 1: [0% grade] 968
Question 2: Propensity score matching by DuGoff et al. (2014) [50% grade] 970
Question 3: Propensity score matching by Austin et al. (2018) [50% grade] 971

IX Machine learning (ML) **973**
Background 974
Overview of tutorials 974

Concepts (L) **980**
Machine learning 980
Reading list 980
Video Lessons 980
Video Lesson Slides 981
Links 981
References 981

Continuous outcome **982**

Data spliting **997**

Cross-validation **1002**

Binary outcome **1007**

Supervised learning **1020**

Unsupervised learning **1040**

NHIS Example **1046**

Replicate Results **1064**

Quiz (L) **1094**
Live Quiz 1094
Download Quiz 1094

R functions (L) **1096**

X ML in causal inference **1097**
Background 1098
Overview of tutorials 1098

Concepts (C) **1101**
ML in causal inference 1101
Reading list 1101
Video Lessons 1102
Links 1102
References 1102

Motivation **1103**

SuperLearner **1108**

Binary Outcomes **1114**

Continuous Outcomes **1121**

Comparing results **1128**

Quiz (C) **1154**
Live Quiz 1154
Download Quiz 1154

R functions (C) **1156**

XI Complex outcomes **1157**
Background 1158
Overview of tutorials 1158

Concepts (N) **1161**
Complex outcomes beyond binary 1161
Reading list 1161
Video Lessons 1161

Video Lesson Slides 1162
Links 1162
References 1162

Polytomous and ordinal **1163**

Survival analysis **1174**
Data and Variables 1187

Survival analysis: NHANES **1193**

Poisson **1209**

R functions (N) **1218**

Quiz (N) **1219**
Live Quiz 1219
Download Quiz 1219

XII Longitudinal data **1221**
Background 1222
Overview of tutorials 1222

Concepts (T) **1225**
Longitudinal Data Analysis 1225
Reading list 1225
Video Lessons 1226
Video Lesson Slides 1226
Links 1226
References 1226

Mixed effects models **1227**

GEE **1242**

R functions (T) **1260**

Quiz (T) **1261**
Live Quiz 1261
Download Quiz 1261

XIIIMediation analysis **1263**
Background 1264
Overview of tutorials 1264

Concepts (I) **1267**
Mediation Analysis 1267
Reading list 1267
Video Lessons 1267
Video Lesson Slides 1268
Links 1268
References 1268

Baron and Kenny **1269**

Justification **1280**

Mediation Example **1285**

R functions (I) **1300**

Quiz (I) **1301**
Live Quiz 1301
Download Quiz 1301

XIVWriting tools **1303**
Background 1304
Overview of tutorials 1304

Collaborative Writing **1306**

Formatting Tools **1309**

The Project

Welcome to a website designed to bridge a unique gap in the health research world. This platform specifically targets the complex intersection of health research and advanced statistics; a niche often perceived as challenging by many newcomers. Whether you're taking your first steps into health research or you're grappling with the intricacies of advanced statistical methods, you are in the right place. Here, we offer:

- Tutorials on using the R software, starting from the basics.
- Guides on tapping into genuine survey data from reputable sources like Statistics Canada and the US government's Centers for Disease Control (CDC).
- Step-by-step walkthroughs on conducting and reporting comprehensive epidemiological studies.

This hub is a part of an open educational initiative, meaning it is available to everyone. We hope to raise the standard of health research methodology through this endeavor.

What We Aim to Achieve

We are on a mission to:

- Equip public health learners with hands-on experience.
- Teach the nuances of applying advanced epidemiological methods using real data.
- Offer a unique open textbook that's enriched with interactive tools and quizzes for a self-paced learning experience.

Dive into Our Modules

Embark on a journey through

- 1 introductory module about R (indicator W),
- 9 core learning modules (letters in the parentheses: A, Q, R, P, D, M, S, L, C are the chapter indicators), and
- 4 bonus modules, with N, T, I, G being bonus chapter indicators.

Indicators are listed along with quizzes, R functions, and exercises associated with the corresponding chapters. Only key chapters have exercises (W, A, D, M, S).

Module	Topics.Indicators	Descriptions
1	R for Data Wrangling (W)	Get to know R.
2	Accessing (A) Survey Data Resources	Understand and source reliable national surve
3	Crafting Analytic Data for Research Questions (Q)	Customize data to your research query.
4	Causal Roles (R)	Delve into the concept of confounding and its
5	Predictive (P) modeling	Introduction to key concepts of prediction mo
6	Complex Survey Data (D) Analysis	Handle data sets obtained from complex surve
7	Missing (M) Data Analysis	Understand and tackle missingness in your da
8	Propensity Score (S) Analysis	Dive deeper into advanced observational data
9	Machine Learning (L)	Introduction to machine learning algorithms, a
10	Intergrating Machine Learners in Causal (C) Inference	Discusses the potential pitfalls and challenges
11	Non-binary Outcomes (N)	Statistical techniques to deal with complex or
12	Longitudinal Analysis (T)	Longitudinal data analysis techniques
13	Mediation Analysis (I)	Mediation: decomposing the total effect
14	Scientific Writing Tools (G)	Tools and guides for scientific writing and coll

Note

The tutorial is designed with a consistent structure across all chapters to provide a cohesive and thorough learning experience. Here is what you can expect in each chapter:

1. **Overview**: The first page of each chapter offers a concise summary that outlines the key learning objectives, topics covered, and what you can expect to gain from the chapter. The overview page will

also feature links to the data sources used in the tutorials as well as a form where you can report any bugs or issues you encounter. This helps you quickly grasp the chapter's essence and set learning expectations.

2. **Concepts**: Selected core chapters will include a concept page, where materials (e.g., slides, video lessons, additional FAQs where available) will be included. All of the videos linked here (for lessons or labs) are hosted in YouTube, where users can automatically generate subtitles and captions.

3. **Tutorial topics**: Immediately following the overview/concepts, you will find in-depth tutorials that cover each topic in detail. These are designed to provide comprehensive insights and are spread across multiple pages for easier navigation and understanding.

4. **Summary of R functions**: Each chapter includes a succinct summary of the R functions used in the tutorials. This serves as a quick reference guide for learners to understand the tools they will be applying.

5. **Chapter-specific quiz**: For those interested in self-assessment, each chapter concludes with an optional quiz. This is a self-paced learning tool to help reinforce the chapter's key concepts.

6. **Web-App**: A few chapters include shiny apps. Users can work with these apps directly from this website, or will have the option to download and run the app locally.

7. **Practice exercises**: Finally, practice exercises are available for selected chapters to help you apply what you have learned in a hands-on manner. These exercises are designed to reinforce your understanding and give you practical experience with the chapter's topics. Some of these practice exercises may be

> used in future versions of the course, so you may see references to submitting assignments or the points value of a question in an assignment.

How Our Content is Presented

All our resources are hosted on an easy-to-access GitHub page. The format? Engaging text, reproducible software codes, clear analysis outputs, and crisp videos that distill complex topics. And do not miss our quiz section at the end of each module for a quick self-check on what you have learned. This document is created using quarto and R.

The content was primarily designed for a website in HTML format; however, a PDF version based on the website has also been created. Although the formatting is not perfect for this converted PDF, this PDF can be downloaded from here and used for offline reading.

Open Copyright License

CC-BY 4.0

Grant Applicants

Dive into this captivating content, brought to life with the generous support of the UBC OER Fund Implementation Grant and further supported by UBC SPPH. The foundation of this content traces back to the PI's work over five years while instructing SPPH 604 (2018-2022). That knowledge has now been transformed into an open educational resource, thanks to this grant. Meet the innovative minds behind the grant proposal below.

Role	Team_Member	Affiliation
Principal Applicant (PI)	Dr. M Ehsan Karim	UBC School of Population and Public Health
Co-applicant (Co-I)	Dr. Suborna Ahmed	UBC Department of Forest Resources Management
Trainee co-applicants	Md Belal Hossain	UBC School of Population and Public Health
	Fardowsa Yusuf	UBC School of Population and Public Health
	Hanna Frank	UBC School of Population and Public Health
	Dr. Michael Asamoah-Boaheng	UBC Department of Emergency Medicine
	Chuyi (Astra) Zheng	UBC Faculty of Arts

A presentation about the output of this grant in the *OER Project Virtual Showcase and Poster Session* (March 7, 2024):

Toggle Show/Hide

Acknowledgements

We also want to acknowledge earlier contributors to the course material development, who were not part of this current OER grant, include Derek Ouyang, Kate McLeod (both from UBC School of Population and Public Health), and Mohammad Atiquzzaman (UBC Pharmaceutical Sciences). Numerous pieces of student feedback were also incorporated in order to update the content.

How to Cite

Style	Citation
APA	Karim M. E., Epi-OER team (2024). Advanced Epidemiological Methods . Retrieved from https:/
MLA	Karim M. E., Epi-OER team . " Advanced Epidemiological Methods ." Web. April 20, 2024 < https
Chicago	Karim M. E., Epi-OER team . " Advanced Epidemiological Methods ." 2024 . Web. April 20, 2024
Harvard	Karim M. E., Epi-OER team (2024) ' Advanced Epidemiological Methods '. Available at: https://
Vancouver	Karim M. E., Epi-OER team . Advanced Epidemiological Methods . 2024 . [Online]. Available at: h
IEEE	Karim M. E., Epi-OER team , " Advanced Epidemiological Methods ," 2024 , [Online]. Available: h
AMA	Karim M. E., Epi-OER team Advanced Epidemiological Methods . 2024 . [Online]. Available at: ht

- **Epi-OER team**: Hossain MB, Frank HA, Yusuf FL, Ahmed SS, Asamoah-Boaheng M, Zheng C (team is listed in order of contribution to the creation of this book)

The BibTex format can be downloaded from here.

Part I

Data wrangling

Background

The realm of data science is vast, and one of its foundational pillars is data wrangling. Data wrangling, often known as data munging, is the process of transforming raw data into a more digestible and usable format for analysis. In the context of R, a powerful statistical programming language, data wrangling becomes an essential skill for any data enthusiast. This chapter is dedicated to imparting practical knowledge on various data manipulation, import, and summarization techniques in R. Through a series of meticulously crafted tutorials, you will be equipped with the tools and techniques to handle, transform, and visualize data efficiently.

In this chapter, we embark on a structured journey through the intricate world of data wrangling in R. We begin by laying a solid foundation with R Basics, ensuring you grasp the essential elements of R programming. Once grounded in the basics, we progress to understanding the core R Data Types, diving deep into matrices, lists, and data frames. With a firm grasp of these structures, we introduce Automating Tasks to empower you with techniques that streamline the handling of vast datasets. Following this, we delve into the practical aspects of Importing Datasets, showcasing various methods to bring data from different formats into R. Building on this, Data Manipulation comes next, where we explore the myriad ways to modify and reshape your datasets to suit analytical needs. We then turn our attention to Importing External Data, offering a hands-on demonstration of how to integrate specific external datasets into your R environment. As we approach the chapter's culmination, we emphasize the importance of Summary Tables in medical research, teaching you the art and science of data summarization. Finally, we wrap up with R Markdown, providing a comprehensive guide on how to seamlessly document your R code and analytical findings, ensuring your work is both reproducible and presentable.

Introducing R Basics at the outset of an epidemiological methods tutorial book is akin to laying the foundation before building a house. It ensures that all readers, regardless of their prior experience, start on the same page, understanding the fundamental tools and language of R. This foundational knowledge not only smoothens the learning curve but also boosts confidence, allowing learners to focus on complex epidemiological techniques without being bogged down by the intricacies of the R language. In essence, mastering the basics first ensures a more cohesive and effective learning experience as the material advances.

> **!** Important
>
> **Datasets**:
> All of the datasets used in this tutorial can be accessed from this GitHub repository folder

Overview of tutorials

R Basics

This tutorial introduces the basics of R programming. It covers topics such as setting up R and RStudio, using R as a calculator, creating variables, working with vectors, plotting data, and accessing help resources.

R Data Types

This tutorial covers three primary data structures in R: matrices, lists, and data frames. Matrices are two-dimensional arrays with elements of the same type, and their manipulation includes reshaping and combining. Lists in R are versatile collections that can store various R objects, including matrices. Data frames, on the other hand, are akin to matrices but permit columns of diverse data types. The tutorial offers guidance on creating, modifying, and merging data frames and checking their dimensions.

Automating Tasks

Medical data analysis often grapples with vast and intricate data sets. Manual handling isn't just tedious; it's error-prone, especially given the critical decisions hinging on the results. This tutorial introduces automation techniques in R, a leading language for statistical analysis. By learning to use loops and functions, you can automate repetitive tasks, minimize errors, and conduct analyses more efficiently. Dive in to enhance your data handling skills.

Importing Dataset from the Local Computer

This tutorial focuses on importing data into R. It demonstrates how to import data from CSV and SAS formats using functions like read.csv and sasxport.get. It also includes examples of loading specific variables, dropping variables, subsetting observations based on certain criteria, and handling missing values.

Data Manipulation

This tutorial explores various data manipulation techniques in R. It covers topics such as dropping variables from a dataset, keeping specific variables, subsetting observations based on specific criteria, converting variable types (e.g., factors, strings), and handling missing values.

Import External Data Over the Internet

This tutorial provides examples of importing external data into R. It includes specific examples of importing a CSV file (Employee Salaries - 2017 data) and a SAS file (NHANES 2015-2016 data). It also demonstrates how to save a working dataset in different formats, such as CSV and RData.

Summary Tables

This tutorial emphasizes the importance of data summarization in medical research and epidemiology, specifically how to summarize medical data using R. It demonstrates creating "Table 1", a typical descriptive statistics table in research papers, with examples that use the built-in R functions and specialized packages to efficiently summarize and stratify data.

R Markdown

This beginner-friendly tutorial guides you through working with R Markdown (RMD) files in RStudio, a popular IDE for R. The tutorial covers installing prerequisites, creating a new RMD file, and the basics of "knitting" to compile the document into various formats like HTML, PDF, or Word. It delves into embedding R code chunks and plain text within the RMD file, using the knitr package for document rendering. Tips for troubleshooting common issues and additional resources for further learning are also provided.

Tip

Optional Content:
You'll find that some sections conclude with an optional video walkthrough that demonstrates the code. Keep in mind that the content might have been updated since these videos were recorded. Watching these videos is optional.

Warning

Bug Report:
Fill out this form to report any issues with the tutorial.

R basics

Start using R

To get started with R, follow these steps:

- Download and Install R: Grab the newest version from the official R website. > *Tip: Download from a Comprehensive R Archive Network (CRAN) server near your geographic location.*
- Download and Install RStudio: You can get it from this link. > *Note: RStudio serves as an Integrated Development Environment (IDE) offering a user-friendly interface. It facilitates operations such as executing R commands, preserving scripts, inspecting results, managing data, and more.*
- Begin with RStudio: Once you open RStudio, delve into using R. For starters, employ the R syntax for script preservation, allowing future code adjustments and additions.

Basic syntax

> **Tip**
>
> R, a versatile programming language for statistics and data analysis, can execute numerous tasks. Let's break down some of the fundamental aspects of R's syntax.

1. Using R as a Calculator

Similar to how you'd use a traditional calculator for basic arithmetic operations, R can perform these functions with ease. For instance:

```
# Simple arithmetic
1 + 1
#> [1] 2
```

This is a basic addition, resulting in 2.

A more intricate calculation:

```
# Complex calculation involving
# multiplication, subtraction, division, powers, and square root
20 * 5 - 10 * (3/4) * (2^3) + sqrt(25)
#> [1] 45
```

This demonstrates R's capability to handle complex arithmetic operations.

2. Variable Assignment in R

R allows you to store values in variables, acting like labeled containers that can be recalled and manipulated later. For example,

```
# Assigning a value of 2 to variable x1
x1 <- 2
print(x1)
#> [1] 2
```

Similarly:

```
x2 <- 9
x2
#> [1] 9
```

3. Creating New Variables Using Existing Ones

You can combine and manipulate previously assigned variables to create new ones.

```r
# Using variable x1
# to compute its square and assign to y1
y1 <- x1^2
y1
#> [1] 4
```

You can also use multiple variables in a single expression:

```r
y2 <- 310 - x1 + 2*x2 - 5*y1^3
y2
#> [1] 6
```

4. Creating Functions

Functions act as reusable blocks of code. Once defined, they can be called multiple times with different arguments. Here's how to define a function that squares a number:

```r
z <- function(x) {x^2}
```

Call the function

```r
z(x = 0.5)
#> [1] 0.25
z(x = 2)
#> [1] 4
```

R also comes with a plethora of built-in functions. Examples include `exp` (exponential function) and `rnorm` (random number generation from a normal distribution).

5. Utilizing Built-In Functions

For instance, using the exponential function:

```r
# Calling functions
exp(x1)
#> [1] 7.389056
log(exp(x1))
#> [1] 2
```

The `rnorm` function can generate random samples from a normal distribution: below we are generating 10 random sampling from the normal distribution with mean 0 and standard deviation 1:

```
rnorm(n = 10, mean = 0, sd = 1)
#>  [1]  0.3226166 -1.9060127 -0.1138359 -0.6969304  0.9645166  1.1095097
#>  [7]  2.0302084  1.6796770 -0.3809382  0.4759079
```

As random number generation relies on algorithms, results will differ with each execution.

```
# Random sampling (again)
rnorm(n = 10, mean = 0, sd = 1)
#>  [1] -0.07912430  0.50624205  0.78282357 -0.87198802 -0.17179388  1.82821480
#>  [7] -0.01026682  1.51766942  1.08685542 -0.67036414
```

However, by setting a `seed`, we can reproduce identical random results:

```
# Random sampling (again, but with a seed)
set.seed(11)
rnorm(n = 10, mean = 0, sd = 1)
#>  [1] -0.59103110  0.02659437 -1.51655310 -1.36265335  1.17848916 -0.93415132
#>  [7]  1.32360565  0.62491779 -0.04572296 -1.00412058
```

```
# random sampling (reproducing the same numbers)
set.seed(11)
rnorm(n = 10, mean = 0, sd = 1)
#>  [1] -0.59103110  0.02659437 -1.51655310 -1.36265335  1.17848916 -0.93415132
#>  [7]  1.32360565  0.62491779 -0.04572296 -1.00412058
```

As we can see, when we set the same seed, we get exactly the same random number. This is very important for reproducing the same results. There are many other pre-existing functions in R.

6. Seeking Help in R

> **Tip**
>
> R's help function, invoked with `?function_name`, provides detailed documentation on functions, assisting users with unclear or forgotten arguments:

```r
# Searching for help if you know
# the exact name of the function with a question mark
?curve
```

Below is an example of using the pre-exiting function for plotting a curve ranging from -10 to 10.

```r
# Plotting a function
curve(z, from = -10, to = 10, xlab = "x", ylab = "Squared x")
```

If some of the arguments are difficult to remember or what else could be done with that function, we could use the `help` function. For example, we can simply type `help(curve)` or `?curve` to get help on the `curve` function:

> **Tip**
>
> If you're uncertain about a function's precise name, two question marks can assist in the search:

```r
# Searching for help if don't know
# the exact name of the function
??boxplot
```

7. Creating Vectors

Vectors are sequences of data elements of the same basic type. Here are some methods to create them:

```r
# Creating vectors in different ways
x3 <- c(1, 2, 3, 4, 5)
print(x3)
#> [1] 1 2 3 4 5

x4 <- 1:7
print(x4)
#> [1] 1 2 3 4 5 6 7

x5 <- seq(from = 0, to = 100, by = 10)
print(x5)
#>  [1]   0  10  20  30  40  50  60  70  80  90 100

x6 <- seq(10, 30, length = 7)
x6
#> [1] 10.00000 13.33333 16.66667 20.00000 23.33333 26.66667 30.00000
```

8. Plotting in R

R provides numerous plotting capabilities. For instance, the plot function can create scatter plots and line graphs:

```r
# Scatter plot
plot(x5, type = "p", main = "Scatter plot")
```

```r
# Line graph
plot(x = x6, y = x6^2, type = "l", main = "Line graph")
```

9. Character Vectors Apart from numeric values, R also allows for character vectors. For example, we can create a `sex` variable coded as females, males and other.

```r
# Character vector
sex <- c("females", "males", "other")
```

```r
sex
#> [1] "females" "males"   "other"
```

To determine a variable's type, use the mode function:

```r
# Check data type
mode(sex)
#> [1] "character"
```

Package Management

Packages in R are collections of functions and datasets developed by the community. They enhance the capability of R by adding new functions for data analysis, visualization, data import, and more. Understanding how to install and load packages is essential for effective R programming.

1. Installing Packages from CRAN

The CRAN is a major source of R packages. You can install them directly from within R using the `install.packages()` function.

```r
# Installing the 'ggplot2' package
install.packages("ggplot2")
```

2. Loading a Package

After a package is installed, it must be loaded to use its functions. This is done with the `library()` function.

```r
# Loading the 'ggplot2' package
library(ggplot2)
```

You only need to install a package once, but you'll need to load it every time you start a new R session and want to use its functions.

3. Updating Packages

R packages are frequently updated. To ensure you have the latest version of a package, use the `update.packages()` function.

```
# Updating all installed packages
# could be time consuming!
update.packages(ask = FALSE)
# 'ask = FALSE' updates all without asking for confirmation
```

4. Listing Installed Packages

You can view all the installed packages on your R setup using the installed.packages() function.

```
# Listing installed packages
installed.packages()[, "Package"]
```

5. Removing a Package

If you no longer need a package, it can be removed using the `remove.packages()` function.

```
# Removing the 'ggplot2' package
remove.packages("ggplot2")
```

6. Installing Packages from Other Sources

While CRAN is the primary source, sometimes you might need to install packages from GitHub or other repositories. The `devtools` package provides a function for this.

```
# Installing devtools first
install.packages("devtools")
# Loading devtools
library(devtools)
# Install a package from GitHub
# https://github.com/ehsanx/simMSM
install_github("ehsanx/simMSM")
```

When you are working on a project, it's a good practice to list and install required packages at the beginning of your R script.

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Data types

Matrix

> **Tip**
>
> In R, matrices are two-dimensional rectangular data sets, which can be created using the `matrix()` function. It's essential to remember that all the elements of a matrix must be of the same type, such as all numeric or all character.

To construct a matrix, we often start with a vector and specify how we want to reshape it. For instance:

```r
# Matrix 1
x <- 1:10
matrix1 <- matrix(x, nrow = 5, ncol = 2, byrow = TRUE)
matrix1
#>      [,1] [,2]
#> [1,]    1    2
#> [2,]    3    4
#> [3,]    5    6
#> [4,]    7    8
#> [5,]    9   10
```

Here, the vector x contains numbers from 1 to 10. We reshape it into a matrix with 5 rows and 2 columns. The `byrow = TRUE` argument means the matrix will be filled row-wise, with numbers from the vector.

Conversely, if you want the matrix to be filled column-wise, you'd set `byrow = FALSE`:

```r
# matrix 2
matrix2 <- matrix(x, nrow = 5, ncol = 2, byrow = FALSE)
matrix2
#>      [,1] [,2]
#> [1,]    1    6
#> [2,]    2    7
#> [3,]    3    8
#> [4,]    4    9
#> [5,]    5   10
```

You can also combine or concatenate matrices. `cbind()` joins matrices by columns while `rbind()` joins them by rows.

```r
# Merging 2 matrices
cbind(matrix1, matrix2)
#>      [,1] [,2] [,3] [,4]
#> [1,]    1    2    1    6
#> [2,]    3    4    2    7
#> [3,]    5    6    3    8
#> [4,]    7    8    4    9
#> [5,]    9   10    5   10
```

```r
# Appending 2 matrices
rbind(matrix1, matrix2)
#>       [,1] [,2]
#>  [1,]    1    2
#>  [2,]    3    4
#>  [3,]    5    6
#>  [4,]    7    8
#>  [5,]    9   10
#>  [6,]    1    6
#>  [7,]    2    7
#>  [8,]    3    8
#>  [9,]    4    9
#> [10,]    5   10
```

Creating an empty matrix is also possible:

```r
matrix(nrow=5, ncol=5)
#>      [,1] [,2] [,3] [,4] [,5]
#> [1,]   NA   NA   NA   NA   NA
#> [2,]   NA   NA   NA   NA   NA
#> [3,]   NA   NA   NA   NA   NA
#> [4,]   NA   NA   NA   NA   NA
#> [5,]   NA   NA   NA   NA   NA
```

List

> **Tip**
>
> In R, lists can be seen as a collection where you can store a variety of different objects under a single name. This includes vectors, matrices, or even other lists. It's very versatile because its components can be of any type of R object, such as vector, matrix, array, dataframe, table, list, and so on.

For instance:

```r
# List of 2 matrices
list1 <- list(matrix1, matrix2)
list1
#> [[1]]
#>      [,1] [,2]
#> [1,]    1    2
#> [2,]    3    4
#> [3,]    5    6
#> [4,]    7    8
#> [5,]    9   10
#>
#> [[2]]
#>      [,1] [,2]
#> [1,]    1    6
#> [2,]    2    7
#> [3,]    3    8
#> [4,]    4    9
#> [5,]    5   10
```

Lists can also be expanded to include multiple items:

```
x6 <- seq(10, 30, length = 7)
sex <- c("females", "males", "other")
# Expanding list to include more items
list2 <- list(list1, x6, sex, matrix1)
list2
#> [[1]]
#> [[1]][[1]]
#>      [,1] [,2]
#> [1,]    1    2
#> [2,]    3    4
#> [3,]    5    6
#> [4,]    7    8
#> [5,]    9   10
#>
#> [[1]][[2]]
#>      [,1] [,2]
#> [1,]    1    6
#> [2,]    2    7
#> [3,]    3    8
#> [4,]    4    9
#> [5,]    5   10
#>
#>
#> [[2]]
#> [1] 10.00000 13.33333 16.66667 20.00000 23.33333 26.66667 30.00000
#>
#> [[3]]
#> [1] "females" "males"   "other"
#>
#> [[4]]
#>      [,1] [,2]
#> [1,]    1    2
#> [2,]    3    4
#> [3,]    5    6
#> [4,]    7    8
#> [5,]    9   10
```

Combining different types of data into a single matrix converts everything to a character type:

```r
# A matrix with numeric and character variables
id <- c(1, 2)
score <- c(85, 85)
sex <- c("M", "F")
new.matrix <- cbind(id, score, sex)
new.matrix
#>      id  score sex
#> [1,] "1" "85"  "M"
#> [2,] "2" "85"  "F"
```

To check the type of data in your matrix:

```r
mode(new.matrix)
#> [1] "character"
```

Data frame

> **Tip**
>
> As we can see combining both numeric and character variables into a matrix ended up with a matrix of character values. To keep the numeric variables as numeric and character variables as character, we can use the `data.frame` function.

1. Creating a data frame

```r
df <- data.frame(id, score, sex)
df
#>   id score sex
#> 1  1    85   M
#> 2  2    85   F
```

A data frame is similar to a matrix but allows for columns of different types (numeric, character, factor, etc.). It's a standard format for storing data sets in R.

To check the mode or type of your data frame:

```r
mode(df)
#> [1] "list"
```

2. Extract elements

Data frames allow easy extraction and modification of specific elements. For example, we can extract the values on the first row and first column as follow:

```
df[1,1]
#> [1] 1
```

Similarly, the first column can be extracted as follows:

```
df[,1]
#> [1] 1 2
```

The first row can be extracted as follows:

```
df[1,]
#>   id score sex
#> 1  1    85   M
```

Columns can also be accessed using `$`. Below we are calling column `id`:

```
df$id
#> [1] 1 2
```

3. Modifying values

We can edit the values in the data frame as well. For example, we can change the score from 85 to 90 for the id 1:

```
df$score[df$id == 1] <- 90
df
#>   id score sex
#> 1  1    90   M
#> 2  2    85   F
```

We can also change the name of the variables/columns:

```r
colnames(df) <- c("Studyid", "Grade", "Sex")
df
#>   Studyid Grade Sex
#> 1       1    90   M
#> 2       2    85   F
```

4. Combining data frames

We can also merge another data frame with the same variables using the `rbind` function:

```r
# Create a new dataset
df2 <- data.frame(Studyid = c(10, 15, 50), Grade = c(75, 90, 65), Sex = c("F", "M", "M"))

# Combining two data frames
df.new <- rbind(df, df2)

# Print the first 6 rows
head(df.new)
#>   Studyid Grade Sex
#> 1       1    90   M
#> 2       2    85   F
#> 3      10    75   F
#> 4      15    90   M
#> 5      50    65   M
```

5. Checking the dimensions

To see the dimension of the data frame (i.e., number of rows and columns), we can use the `dim` function:

```r
dim(df.new)
#> [1] 5 3
```

As we can see, we have 5 rows and 3 columns. We can use the `nrow` and `ncol` functions respectively for the same output:

```r
nrow(df.new)
#> [1] 5
ncol(df.new)
#> [1] 3
```

Automating tasks

Repeating a task

> **Tip**
>
> The `for` loop is a control flow statement in R that lets you repeat a particular task multiple times. This repetition is based on a sequence of numbers or values in a vector.

Consider a simple real-life analogy: Imagine you are filling water in 10 bottles, one by one. Instead of doing it manually 10 times, you can set a machine to do it in a loop until all 10 bottles are filled.

1. Example 1

Let's initiate a counter k at 0 and add 5 to k with each iteration of the loop (i.e., every time it "runs"). After 10 cycles, the loop will stop, but not before printing k in each cycle.

```r
# Looping and adding
k <- 0
for (i in 1:10){
  k <- k + 5
  print(k)
}
#> [1] 5
#> [1] 10
#> [1] 15
#> [1] 20
#> [1] 25
#> [1] 30
#> [1] 35
#> [1] 40
```

```
#> [1] 45
#> [1] 50
```

2. Example 2

We create a variable `x5` containing the values of 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100. Let us print the first 5 values using the `for` loop function:

```
x5 <- seq(from = 0, to = 100, by = 10)
# Looping through a vector
k <- 1:5
for (ii in k){
  print(x5[ii])
}
#> [1] 0
#> [1] 10
#> [1] 20
#> [1] 30
#> [1] 40
```

This loop cycles through the first five values of a previously created variable `x5` and prints them. Each value printed corresponds to the positions 1 to 5 in `x5`.

3. Example 3

Let us use the `for` loop in a more complicated scenario. First, we create a vector of numeric values and square it:

```
# Create a vector
k <- c(1, 3, 6, 2, 0)
k^2
#> [1]  1  9 36  4  0
```

This is just squaring each value in the vector `k`.

4. Example 4

Using the `for` loop function, we can create the same vector of square values as in Example 3. To do so, (i) we create a null object, (ii) use the loop for each of the elements in the vector (k), (iii) square each of the elements, and (iv) store each of the elements of the new vector. In the example below, the length of k is 5, and the loop will run from the first to the fifth element of k. Also, k.sq[1] is the first stored value for squared-k, and k.sq[2] is the second stored value for squared-k, and so on.

```r
# Looping through a vector with function
k.sq <- NULL
for (i in 1:length(k)){
  k.sq[i] <- k[i]^2
}

# Print the values
k.sq
#> [1]  1  9 36  4  0
```

Here, we achieve the same result as the third example but use a for loop. We prepare an empty object `k.sq` and then use the loop to square each value in `k`, storing the result in `k.sq`.

5. Example 5

```r
df.new <- data.frame(
  Studyid = c(1, 2, 10, 15, 50),
  Grade = c(90, 85, 75, 90, 65),
  Sex = c('M', 'F', 'F', 'M', 'M')
)
# Looping through a data frame
for (i in 1:nrow(df.new)){
  print(df.new[i,"Sex"])
}
#> [1] "M"
#> [1] "F"
#> [1] "F"
#> [1] "M"
#> [1] "M"
```

This loop prints the "Sex" column value for each row in the df.new data frame.

Functions

> **Tip**
>
> A function in R is a piece of code that can take inputs, process them, and return an output. There are functions built into R, like `mean()`, which calculates the average of a set of numbers.

1. Built-in function

```r
# Calculating a mean from a vector
Vector <- 1:100
mean(Vector)
#> [1] 50.5
```

Here, we're using the built-in `mean()` function to find the average of numbers from 1 to 100.

2. Custom-made function

To understand how functions work, sometimes it's helpful to build our own. Now we will create our own function to calculate the mean, where we will use the following equation to calculate it:

Mean $= \frac{\sum_{i=1}^{n} x_i}{n}$,

where x_1, x_2,..., x_n are the values in the vector and n is the sample size. Let us create the function for calculation the mean:

This function, `mean.own`, calculates the average. We add up all the numbers in a vector (`Sum <- sum(x)`) and divide by the number of items in that vector (`n <- length(x)`). The result is then returned.

```r
mean.own <- function(x){
  Sum <- sum(x)
  n <- length(x)
  return(Sum/n)
}
```

By using our custom-made function, we calculate the mean of numbers from 1 to 100, getting the same result as the built-in `mean()` function.

```r
mean.own(Vector)
#> [1] 50.5
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Importing dataset

Introduction to Data Importing

Before analyzing data in R, one of the first steps you'll typically undertake is importing your dataset. R provides numerous methods to do this, depending on the format of your dataset.

Datasets come in a variety of file formats, with .csv (Comma-Separated Values) and .txt (Text file) being among the most common. While R's interface offers manual ways to load these datasets, knowing how to code this step ensures better reproducibility and automation.

Importing .txt files

A .txt data file can be imported using the `read.table` function. As an example, consider you have a dataset named grade in the specified path.

Let's briefly glance at the file without concerning ourselves with its formatting.

```r
# Read and print the content of the TXT file
content <- readLines("Data/wrangling/grade.txt")
cat(content, sep = "\n")
#> Studyid Grade Sex
#> 1    90   M
#> 2    85   F
#> 10    75   F
#> 15    90   M
#> 50    65   M
```

Using the `read.table` function, you can load this dataset in R properly. It's important to specify `header = TRUE` if the first row of your dataset contains variable names.

Tip: Always ensure the `header` argument matches the structure of your dataset. If your dataset contains variable names, set `header = TRUE`.

```r
## Read a text dataset
grade <- read.table("Data/wrangling/grade.txt", header = TRUE)

# Display the first few rows of the dataset
head(grade)
#>   Studyid Grade Sex
#> 1       1    90   M
#> 2       2    85   F
#> 3      10    75   F
#> 4      15    90   M
#> 5      50    65   M
```

Importing .csv files

Similarly, .csv files can be loaded using the `read.csv` function. Here's how you can load a .csv dataset named mpg:

```r
## Read a csv dataset
mpg <- read.csv("Data/wrangling/mpg.csv", header = TRUE)
# Display the first few rows of the dataset
head(mpg)
#>   manufacturer model displ year cyl      trans drv cty hwy fl   class
#> 1         audi    a4   1.8 1999   4   auto(l5)   f  18  29  p compact
#> 2         audi    a4   1.8 1999   4 manual(m5)   f  21  29  p compact
#> 3         audi    a4   2.0 2008   4 manual(m6)   f  20  31  p compact
#> 4         audi    a4   2.0 2008   4   auto(av)   f  21  30  p compact
#> 5         audi    a4   2.8 1999   6   auto(l5)   f  16  26  p compact
#> 6         audi    a4   2.8 1999   6 manual(m5)   f  18  26  p compact
```

While we've discussed two popular data formats, R can handle a plethora of other formats. For further details, refer to Quick-R (2023). Notably, some datasets come built-in with R packages, like the mpg dataset in the ggplot2 package. To load such a dataset:

```r
data(mpg, package = "ggplot2")
head(mpg)
#> # A tibble: 6 x 11
#>   manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
#>   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
#> 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa~
#> 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa~
#> 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa~
#> 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa~
#> 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa~
#> 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa~
```

To understand more about the variables and the dataset's structure, you can consult the documentation:

```r
?mpg
```

Data Screening and Understanding Your Dataset

`dim()`, `nrow()`, `ncol()`, and `str()` are incredibly handy functions when initially exploring your dataset.

Once your data is in R, the next logical step is to get familiar with it. Knowing the dimensions of your dataset, types of variables, and the first few entries can give you a quick sense of what you're dealing with.

For instance, `str` (short for structure) is a concise way to display information about your data. It reveals the type of each variable, the first few entries, and the total number of observations:

```r
str(mpg)
#> tibble [234 x 11] (S3: tbl_df/tbl/data.frame)
#>  $ manufacturer: chr [1:234] "audi" "audi" "audi" "audi" ...
#>  $ model       : chr [1:234] "a4" "a4" "a4" "a4" ...
#>  $ displ       : num [1:234] 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
#>  $ year        : int [1:234] 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
#>  $ cyl         : int [1:234] 4 4 4 4 6 6 6 4 4 4 ...
#>  $ trans       : chr [1:234] "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
```

```
#>  $ drv         : chr [1:234] "f" "f" "f" "f" ...
#>  $ cty         : int [1:234] 18 21 20 21 16 18 18 18 16 20 ...
#>  $ hwy         : int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
#>  $ fl          : chr [1:234] "p" "p" "p" "p" ...
#>  $ class       : chr [1:234] "compact" "compact" "compact" "compact" ...
```

In summary, becoming proficient in data importing and initial screening is a fundamental step in any data analysis process in R. It ensures that subsequent stages of data manipulation and analysis are based on a clear understanding of the dataset at hand.

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

References

Data manipulation

Data manipulation is a foundational skill for data analysis. This guide introduces common methods for subsetting datasets, handling variable types, creating summary tables, and dealing with missing values using R.

Load dataset

Understanding the dataset's structure is the first step in data manipulation. Here, we're using the `mpg` dataset, which provides information on various car models:

```
mpg <- read.csv("Data/wrangling/mpg.csv", header = TRUE)
```

Subset

Often, you'll need to subset your data for analysis. Here, we'll explore different methods to both drop unwanted variables and keep desired observations.

Drop variables

Sometimes, only part of the variables will be used in your analysis. Therefore, you may want to drop the variables you do not need. There are multiple ways to drop variables from a dataset. Below are two examples without using any package and using the `dplyr` package.

> Tip
>
> **Option 1: No package needed**
> dataset.name[, c(columns_names_you_want_to_KEEP)]

Say, we want to keep only three variables in the mpg dataset: manufacturer, model and cyl. For Option 1 (without package), we can use the following R codes to keep these three variables:

```
mpg1 <- mpg[, c("manufacturer", "model", "cyl")]
head(mpg1)
#>   manufacturer model cyl
#> 1         audi    a4   4
#> 2         audi    a4   4
#> 3         audi    a4   4
#> 4         audi    a4   4
#> 5         audi    a4   6
#> 6         audi    a4   6
```

Here `mpg1` is a new dataset containing only three variables (manufacturer, model and cyl).

> Tip
>
> **Option 2: use select in dplyr**
> select(dataset.name, c(columns_names_you_want_to_KEEP))

For Option 2, the `dplyr` package offers the `select` function, which provides a more intuitive way to subset data.

```
mpg2 <- select(mpg, c("manufacturer", "model", "cyl"))
head(mpg2)
#>   manufacturer model cyl
#> 1         audi    a4   4
#> 2         audi    a4   4
#> 3         audi    a4   4
#> 4         audi    a4   4
#> 5         audi    a4   6
#> 6         audi    a4   6
```

We can also exclude any variables from the dataset by using the minus (-) sign with the `select` function. For example, we we want to drop `trans`, `drv`, and `cty` from the `mpg` dataset, we can use the following codes:

```
mpg3 <- select(mpg, -c("trans", "drv", "cty"))
head(mpg3)
#>   manufacturer model displ year cyl hwy fl   class
#> 1         audi    a4   1.8 1999   4  29  p compact
#> 2         audi    a4   1.8 1999   4  29  p compact
#> 3         audi    a4   2.0 2008   4  31  p compact
#> 4         audi    a4   2.0 2008   4  30  p compact
#> 5         audi    a4   2.8 1999   6  26  p compact
#> 6         audi    a4   2.8 1999   6  26  p compact
```

This `mpg3` is a new dataset from `mpg` after dropping three variables (trans, drv, and cty).

Keep observations

It often happens that we only want to investigate a subset of a population which only requires a subset of our dataset. In this case, we need to subset the dataset to meet certain requirements. Again, there are multiple ways to do this task. Below is an example without a package and with the `dplyr` package:

> Tip
>
> **Option 1: No package needed**
> dataset.name[rows_you_want_to_KEEP,]

> Tip
>
> **Option 2: No package needed**
> subset(dataset.name, rows_you_want_to_KEEP)

Tip

Option 3: use filter in dplyr
filter(dataset.name, rows_you_want_to_KEEP)

We can use the logical tests for the rows you want to keep or drop.

Tip

Common logical tests are:

Syntax	Meaning
X <(=) Y	Smaller (equal) than
X >(=) Y	Larger (equal) than
X == Y	Equal to
X != Y	Not equal to
is.na(X)	is NA/missing?

Say, we want to keep the observations for which cars are manufactured in 2008. We can use the following R codes to do it:

```
# Option 1
mpg4 <- mpg[mpg$year == "2008",]
head(mpg4)
#>    manufacturer      model displ year cyl      trans drv cty hwy fl   class
#> 3          audi         a4   2.0 2008   4 manual(m6)   f  20  31  p compact
#> 4          audi         a4   2.0 2008   4   auto(av)   f  21  30  p compact
#> 7          audi         a4   3.1 2008   6   auto(av)   f  18  27  p compact
#> 10         audi a4 quattro   2.0 2008   4 manual(m6)   4  20  28  p compact
#> 11         audi a4 quattro   2.0 2008   4   auto(s6)   4  19  27  p compact
#> 14         audi a4 quattro   3.1 2008   6   auto(s6)   4  17  25  p compact
```

The following codes with the `subset` and `filter` function will do the same:

```
# Option 2
mpg5 <- subset(mpg, year == "2008")
head(mpg5)
```

```
#>    manufacturer      model displ year cyl      trans drv cty hwy fl   class
#> 3          audi         a4   2.0 2008   4 manual(m6)   f  20  31  p compact
#> 4          audi         a4   2.0 2008   4   auto(av)   f  21  30  p compact
#> 7          audi         a4   3.1 2008   6   auto(av)   f  18  27  p compact
#> 10         audi a4 quattro   2.0 2008   4 manual(m6)   4  20  28  p compact
#> 11         audi a4 quattro   2.0 2008   4   auto(s6)   4  19  27  p compact
#> 14         audi a4 quattro   3.1 2008   6   auto(s6)   4  17  25  p compact
```

```r
# Option 3
mpg6 <- filter(mpg, year == "2008")
head(mpg6)
#>   manufacturer      model displ year cyl      trans drv cty hwy fl   class
#> 1         audi         a4   2.0 2008   4 manual(m6)   f  20  31  p compact
#> 2         audi         a4   2.0 2008   4   auto(av)   f  21  30  p compact
#> 3         audi         a4   3.1 2008   6   auto(av)   f  18  27  p compact
#> 4         audi a4 quattro   2.0 2008   4 manual(m6)   4  20  28  p compact
#> 5         audi a4 quattro   2.0 2008   4   auto(s6)   4  19  27  p compact
#> 6         audi a4 quattro   3.1 2008   6   auto(s6)   4  17  25  p compact
```

The filter function can also work when you have multiple criteria (i.e., multiple logical tests) to satisfy. Here, we need Boolean operators to connect different logical tests.

> **Tip**
>
> Common boolean operators are:
>
> | Syntax | Meaning |
> |---|---|
> | & | and |
> | \| | or |
> | ! | not |
> | == | equals to |
> | != | not equal to |
> | > | greater than |
> | < | less than |
> | >= | greater than or equal to |
> | <= | less than or equal to |

Say, we want to keep the observations for 6 and 8 cylinders

(cyl) and engine displacement (displ) greater than or equal to 4 litres. We can use the following codes to do the task:

```
mpg7 <- filter(mpg, cyl %in% c("6","8") & displ >= 4)
head(mpg7)
#>   manufacturer              model displ year cyl    trans drv cty hwy fl
#> 1         audi         a6 quattro   4.2 2008   8 auto(s6)   4  16  23  p
#> 2    chevrolet c1500 suburban 2wd   5.3 2008   8 auto(l4)   r  14  20  r
#> 3    chevrolet c1500 suburban 2wd   5.3 2008   8 auto(l4)   r  11  15  e
#> 4    chevrolet c1500 suburban 2wd   5.3 2008   8 auto(l4)   r  14  20  r
#> 5    chevrolet c1500 suburban 2wd   5.7 1999   8 auto(l4)   r  13  17  r
#> 6    chevrolet c1500 suburban 2wd   6.0 2008   8 auto(l4)   r  12  17  r
#>     class
#> 1 midsize
#> 2     suv
#> 3     suv
#> 4     suv
#> 5     suv
#> 6     suv
```

The `%in%` operator is used to determine whether the values of the first argument are present in the second argument.

Handling Variable Types

> **Tip**
>
> Most common types of variable in R are
>
> - numbers,
> - factors and
> - strings(or character).
>
> Understanding and manipulating these types are crucial for data analysis.

1. Identifying Variable Type

When we analyze the data, we usually just deal with numbers and factors. If there are variables are strings, we could convert them to factors using **as.factors(variable.name)**

```r
mode(mpg$trans)
#> [1] "character"
```

```r
str(mpg$trans)
#>  chr [1:234] "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" "auto(l5)" ...
```

2. Converting Characters to Factors

Sometimes, it's necessary to treat text data as categorical by converting them into factors. **as.numeric()** converts other types of variables to numbers. For a factor variable, we usually we want to access the categories (or levels) it has. We can use a build-in function to explore: **levels(variable.name)**

```r
# no levels for character
levels(mpg$trans)
#> NULL
```

```r
## Ex check how many different trans the dataset has
mpg$trans <- as.factor(mpg$trans)
levels(mpg$trans)
#>  [1] "auto(av)"   "auto(l3)"   "auto(l4)"   "auto(l5)"   "auto(l6)"
#>  [6] "auto(s4)"   "auto(s5)"   "auto(s6)"   "manual(m5)" "manual(m6)"
```

The levels usually will be ordered alphabetically. The first level is called "baseline". However, the users may/may not want to keep this baseline and want to relevel/change the reference group. We can do it using the `relevel` function:

relevel(variable.name, ref=)

```r
mpg$trans <- relevel(mpg$trans, ref = "auto(s6)")
levels(mpg$trans)
#>  [1] "auto(s6)"   "auto(av)"   "auto(l3)"   "auto(l4)"   "auto(l5)"
#>  [6] "auto(l6)"   "auto(s4)"   "auto(s5)"   "manual(m5)" "manual(m6)"
nlevels(mpg$trans)
#> [1] 10
```

The `factor` function can also be used to combine multiple factors into one factor.

```r
## EX re-group trans to "auto" and "manual"
levels(mpg$trans) <- list(auto = c("auto(av)", "auto(l3)", "auto(l4)", "auto(l5)", "auto(l6)",
                                   "auto(s4)", "auto(s5)", "auto(s6)"),
                          manual = c("manual(m5)", "manual(m6)"))
levels(mpg$trans)
#> [1] "auto"   "manual"
```

You can also change the order of all factors using the following code: **factor(variable.name, levels = c("new order"))**

```r
## EX. Change the order of trans to manual
mpg$trans <- factor(mpg$trans, levels = c("manual", "auto"))
levels(mpg$trans)
#> [1] "manual" "auto"
```

3. Converting back from Factors Characters

In R, the use of factors with multiple levels is primarily a memory optimization strategy. While users may not directly see this, R assigns internal numerical identifiers to each level, which is a more memory-efficient way of handling such data. Unlike some other software packages that generate multiple dummy variables to represent a single variable, R's approach is generally more resource-efficient.

You can also convert factor back to character using the **as.character** function.

```r
# Convert factor back to character
mpg$trans <- as.character(mpg$trans)
levels(mpg$trans) # no levels for character
#> NULL
```

Convert continuous variables to categorical variables

> **Tip**
>
> `ifelse`, `cut`, `recode` all are helpful functions to convert numerical variables to categorical variables.

Let's see the summary of the cty variable first.

```r
summary(mpg$cty)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    9.00   14.00   17.00   16.86   19.00   35.00
```

say, we may want to change continuous 'cty' into groups 0-14, 15-18, and 18-40. Below is an example with the `cut` function.

```
## EX. change the cty into two categories (0,14], (14,18] and (18,40]
mpg$cty.num <- cut(mpg$cty, c(0, 14, 18, 40), right = TRUE)
table(mpg$cty.num)
#>
#>  (0,14] (14,18] (18,40]
#>      73      85      76
```

```
## Try this: do you see a difference?: [0,14), [14,18) and [18,40)
mpg$cty.num2 <- cut(mpg$cty, c(0, 14, 18, 40), right = FALSE)
table(mpg$cty.num2)
#>
#>  [0,14) [14,18) [18,40)
#>      54      78     102
```

] stands for closed interval, i.e., right = TRUE. On the other hand,) means open interval. Hence, there will be a huge difference when setting right = TRUE vs. right = FALSE

Missing value

> **Tip**
>
> Incomplete datasets can distort analysis. Identifying and managing these missing values is thus crucial.

We can check how many missing values we have by: **table(is.na(variable.name))**

Let's us check whether the cty variable contains any missing values:

```
table(is.na(mpg$cty))
#>
#> FALSE
#>   234
```

If you want to return all non-missing values, i.e., complete case values: **na.omit(variable.name)**. For more extensive methods on handling missing values, see subsequent tutorials.

Import external data

```r
# Load required packages
library(dplyr)
require(Hmisc)
```

When dealing with data analysis in R, it's common to need to import external data. This tutorial will walk you through importing data in different formats.

CSV format data

CSV stands for "Comma-Separated Values" and it's a widely used format for data. We'll be looking at the "Employee Salaries - 2017" dataset, which contains salary information for permanent employees of Montgomery County in 2017.

Employee Salaries - 2017 data

> **Tip**
>
> We'll be loading the `Employee_Salaries_-_2017.csv` dataset into R from its saved location at `Data/wrangling/`. Do note, the directory path might vary for you based on where you've stored the downloaded data.

```r
data.download <- read.csv("Data/wrangling/Employee_Salaries_-_2017.csv")
```

Here, the `read.csv` function reads the data from the CSV file and stores it in a variable called `data.download`.

To understand the structure of our dataset, We can see the number of rows and columns and the names of the columns/variables as follows:

```r
dim(data.download) # check dimension / row / column numbers
#> [1] 9398   12
nrow(data.download) # check row numbers
#> [1] 9398
names(data.download) # check column names
#>  [1] "Full.Name"                 "Gender"
#>  [3] "Current.Annual.Salary"     "X2017.Gross.Pay.Received"
#>  [5] "X2017.Overtime.Pay"        "Department"
#>  [7] "Department.Name"           "Division"
#>  [9] "Assignment.Category"       "Employee.Position.Title"
#> [11] "Position.Under.Filled"     "Date.First.Hired"
```

> **Tip**
>
> `head` shows the first 6 elements of an object, giving you a sneak peek into the data you're dealing with, while `tail` shows the last 6 elements.

We can see the first see six rows of the dataset as follows:

```r
head(data.download)
#>            Full.Name Gender Current.Annual.Salary X2017.Gross.Pay.Received
#> 1     Aarhus, Pam J.      F              70959.79                 71316.72
#> 2   Aaron, Marsha M.      F             110359.00                108040.82
#> 3 Ababio, Godfred A.      M              55950.24                 62575.19
#> 4     Ababu, Essayas      M              95740.00                 96055.94
#> 5 Abbamonte, Drew B.      M              74732.00                 98736.78
#> 6 Abbasian, Takin M.      M              16451.50                  4547.10
#>   X2017.Overtime.Pay Department                                Department.Name
#> 1               0.00        POL                          Department of Police
#> 2               0.00        HHS     Department of Health and Human Services
#> 3            7649.19        COR                Correction and Rehabilitation
#> 4               0.00        HCA Department of Housing and Community Affairs
#> 5           23468.73        POL                          Department of Police
#> 6              94.92        POL                          Department of Police
#>                                                                   Division
#> 1 MSB Information Mgmt and Tech Division Records Management Section
#> 2                     Adult Protective and Case Management Services
#> 3                                          PRRS Facility and Security
#> 4                                       Affordable Housing Programs
```

```
#> 5                          PSB 6th District Special Assignment Team
#> 6                                                MSB Personnel Division
#>   Assignment.Category     Employee.Position.Title Position.Under.Filled
#> 1    Fulltime-Regular Office Services Coordinator
#> 2    Fulltime-Regular   Supervisory Social Worker
#> 3    Fulltime-Regular      Resident Supervisor II
#> 4    Fulltime-Regular     Planning Specialist III
#> 5    Fulltime-Regular          Police Officer III
#> 6    Parttime-Regular                Police Cadet
#>   Date.First.Hired
#> 1       09/22/1986
#> 2       11/19/1989
#> 3       05/05/2014
#> 4       03/05/2007
#> 5       07/16/2007
#> 6       09/05/2017
```

Next, for learning purposes, let's artificially assign all male genders in our dataset as missing:

```
# Assigning male gender as missing
data.download$Gender[data.download$Gender == "M"] <- NA
head(data.download)
#>            Full.Name Gender Current.Annual.Salary X2017.Gross.Pay.Received
#> 1     Aarhus, Pam J.      F              70959.79                 71316.72
#> 2   Aaron, Marsha M.      F             110359.00                108040.82
#> 3 Ababio, Godfred A.   <NA>              55950.24                 62575.19
#> 4     Ababu, Essayas   <NA>              95740.00                 96055.94
#> 5 Abbamonte, Drew B.   <NA>              74732.00                 98736.78
#> 6 Abbasian, Takin M.   <NA>              16451.50                  4547.10
#>   X2017.Overtime.Pay Department                              Department.Name
#> 1               0.00        POL                        Department of Police
#> 2               0.00        HHS     Department of Health and Human Services
#> 3            7649.19        COR                Correction and Rehabilitation
#> 4               0.00        HCA Department of Housing and Community Affairs
#> 5           23468.73        POL                        Department of Police
#> 6              94.92        POL                        Department of Police
#>                                                                   Division
#> 1 MSB Information Mgmt and Tech Division Records Management Section
#> 2                     Adult Protective and Case Management Services
```

```
#> 3                                        PRRS Facility and Security
#> 4                                       Affordable Housing Programs
#> 5                           PSB 6th District Special Assignment Team
#> 6                                                MSB Personnel Division
#>   Assignment.Category     Employee.Position.Title Position.Under.Filled
#> 1    Fulltime-Regular Office Services Coordinator
#> 2    Fulltime-Regular   Supervisory Social Worker
#> 3    Fulltime-Regular      Resident Supervisor II
#> 4    Fulltime-Regular     Planning Specialist III
#> 5    Fulltime-Regular          Police Officer III
#> 6    Parttime-Regular                Police Cadet
#>   Date.First.Hired
#> 1       09/22/1986
#> 2       11/19/1989
#> 3       05/05/2014
#> 4       03/05/2007
#> 5       07/16/2007
#> 6       09/05/2017
```

This chunk sets the Gender column's value to NA (missing) wherever the gender is "M". This is a form of data manipulation, sometimes used to handle missing or incorrect data. If you want to work with datasets that exclude any missing values:

> **Tip**
>
> `na.omit` and `complete.cases` are useful functions to to create datasets with non-NA values

```
# deleting/dropping missing components
data.download2 <- na.omit(data.download)
head(data.download2)
#>                  Full.Name Gender Current.Annual.Salary
#> 1           Aarhus, Pam J.      F              70959.79
#> 2         Aaron, Marsha M.      F             110359.00
#> 7        Abdalla, Eiman M.      F              63977.00
#> 8   Abdelhamed, Shereen N.      F              53274.00
#> 11 Abdul-Ghani, Hasinah J.      F              50576.93
#> 14           Abebe, Emebet      F              22938.50
#>    X2017.Gross.Pay.Received X2017.Overtime.Pay Department
```

```
#> 1                  71316.72               0.00       POL
#> 2                 108040.82               0.00       HHS
#> 7                  62177.20             184.56       FIN
#> 8                  50549.01            1872.92       POL
#> 11                 51657.55             293.30       POL
#> 14                 22001.04             135.76       HHS
#>                          Department.Name
#> 1                     Department of Police
#> 2  Department of Health and Human Services
#> 7                    Department of Finance
#> 8                     Department of Police
#> 11                    Department of Police
#> 14 Department of Health and Human Services
#>                                                          Division
#> 1  MSB Information Mgmt and Tech Division Records Management Section
#> 2                      Adult Protective and Case Management Services
#> 7                                                 General Accounting
#> 8                                            PSB 2nd District Patrol
#> 11        FSB Traffic Division Automated Traffic Enforcement Section
#> 14                                                Area Health Centers
#>    Assignment.Category     Employee.Position.Title Position.Under.Filled
#> 1     Fulltime-Regular Office Services Coordinator
#> 2     Fulltime-Regular   Supervisory Social Worker
#> 7     Fulltime-Regular      Accountant/Auditor III Accountant/Auditor II
#> 8     Fulltime-Regular          Police Officer III      Police Officer I
#> 11    Fulltime-Regular                 Police Aide
#> 14    Parttime-Regular Community Services Aide III
#>    Date.First.Hired
#> 1        09/22/1986
#> 2        11/19/1989
#> 7        06/27/2016
#> 8        01/09/2017
#> 11       02/05/2007
#> 14       05/01/2017
dim(data.download2)
#> [1] 3806   12
```

Here, na.omit is used to remove rows with any missing values. This can be essential when preparing data for certain analyses.

Alternatively, we could have selected only females to drop all males:

```r
data.download3 <- filter(data.download, Gender != "M")
head(data.download3)
#>                 Full.Name Gender Current.Annual.Salary X2017.Gross.Pay.Received
#> 1          Aarhus, Pam J.      F              70959.79                 71316.72
#> 2        Aaron, Marsha M.      F             110359.00                108040.82
#> 3       Abdalla, Eiman M.      F              63977.00                 62177.20
#> 4  Abdelhamed, Shereen N.      F              53274.00                 50549.01
#> 5 Abdul-Ghani, Hasinah J.      F              50576.93                 51657.55
#> 6           Abebe, Emebet      F              22938.50                 22001.04
#>   X2017.Overtime.Pay Department                        Department.Name
#> 1               0.00        POL                     Department of Police
#> 2               0.00        HHS Department of Health and Human Services
#> 3             184.56        FIN                    Department of Finance
#> 4            1872.92        POL                     Department of Police
#> 5             293.30        POL                     Department of Police
#> 6             135.76        HHS Department of Health and Human Services
#>                                                                Division
#> 1 MSB Information Mgmt and Tech Division Records Management Section
#> 2                     Adult Protective and Case Management Services
#> 3                                                General Accounting
#> 4                                             PSB 2nd District Patrol
#> 5        FSB Traffic Division Automated Traffic Enforcement Section
#> 6                                                Area Health Centers
#>   Assignment.Category     Employee.Position.Title Position.Under.Filled
#> 1    Fulltime-Regular Office Services Coordinator
#> 2    Fulltime-Regular   Supervisory Social Worker
#> 3    Fulltime-Regular      Accountant/Auditor III Accountant/Auditor II
#> 4    Fulltime-Regular          Police Officer III      Police Officer I
#> 5    Fulltime-Regular                  Police Aide
#> 6    Parttime-Regular Community Services Aide III
#>   Date.First.Hired
#> 1       09/22/1986
#> 2       11/19/1989
#> 3       06/27/2016
#> 4       01/09/2017
#> 5       02/05/2007
#> 6       05/01/2017
```

And to check the size of this new dataset:

```
# new dimension / row / column numbers
dim(data.download3)
#> [1] 3806   12
```

SAS format data

> **Tip**
>
> SAS is another data format, commonly used in professional statistics and analytics.

Let's explore importing a SAS dataset. We download a SAS formatted dataset from the CDC website.

```
# Link
x <- "https://wwwn.cdc.gov/Nchs/Nhanes/2015-2016/DEMO_I.XPT"

# Data
NHANES1516data <- sasxport.get(x)
#> Processing SAS dataset DEMO_I     ..

# Check dimension / row / column numbers
dim(NHANES1516data)
#> [1] 9971   47

# Check row numbers
nrow(NHANES1516data)
#> [1] 9971

# Check first 10 names
names(NHANES1516data)[1:10]
#>  [1] "seqn"     "sddsrvyr" "ridstatr" "riagendr" "ridageyr" "ridagemn"
#>  [7] "ridreth1" "ridreth3" "ridexmon" "ridexagm"
```

The `sasxport.get` function retrieves the SAS dataset. The following lines, just like before, help understand its structure.

To analyze some of the data:

```r
table(NHANES1516data$riagendr) # tabulating gender variable
#>
#>    1    2
#> 4892 5079
```

This code creates a frequency table of the `riagendr` variable, which represents gender.

Verify these numbers from CDC website

Summary tables

Medical research and epidemiology often involve large, complex datasets. Data summarization is a vital step that transforms these vast datasets into concise, understandable insights. In medical contexts, these summaries can highlight patterns, indicate data inconsistencies, and guide further research. This tutorial will teach you how to use R to efficiently summarize medical data.

In epidemiology and medical research, "Table 1" typically refers to the first table in a research paper or report that provides descriptive statistics of the study population. It offers a snapshot of the baseline characteristics of the study groups, whether in a cohort study, clinical trial, or any other study design.

```r
# Data
mpg <- read.csv("Data/wrangling/mpg.csv", header = TRUE)

# Frequency table for drv
table(mpg$drv)
#>
#>   4   f   r
#> 103 106  25

# Frequency table for manufacturer
table(mpg$manufacturer)
#>
#>       audi  chevrolet      dodge       ford      honda    hyundai       jeep
#>         18         19         37         25          9         14          8
#> land rover    lincoln    mercury     nissan    pontiac     subaru     toyota
#>          4          3          4         13          5         14         34
#> volkswagen
#>         27

```

```r
## Ex create a summary table between manufacturer and drv
table(mpg$drv, mpg$manufacturer)
#>
#>     audi chevrolet dodge ford honda hyundai jeep land rover lincoln mercury
#>   4   11         4    26   13     0       0    8          4       0       4
#>   f    7         5    11    0     9      14    0          0       0       0
#>   r    0        10     0   12     0       0    0          0       3       0
#>
#>     nissan pontiac subaru toyota volkswagen
#>   4      4       0     14     15          0
#>   f      9       5      0     19         27
#>   r      0       0      0      0          0
```

The first line reads a CSV file. It uses the table() function to generate a contingency table (cross-tabulation) between two categorical variables: `drv` (drive) and `manufacturer`. It essentially counts how many times each combination of drv and manufacturer appears in the dataset.

```r
## Get the percentage summary using prop.table
prop.table(table(mpg$drv, mpg$manufacturer), margin = 2)
#>
#>          audi chevrolet     dodge      ford     honda   hyundai      jeep
#>   4 0.6111111 0.2105263 0.7027027 0.5200000 0.0000000 0.0000000 1.0000000
#>   f 0.3888889 0.2631579 0.2972973 0.0000000 1.0000000 1.0000000 0.0000000
#>   r 0.0000000 0.5263158 0.0000000 0.4800000 0.0000000 0.0000000 0.0000000
#>
#>     land rover   lincoln   mercury    nissan   pontiac    subaru    toyota
#>   4  1.0000000 0.0000000 1.0000000 0.3076923 0.0000000 1.0000000 0.4411765
#>   f  0.0000000 0.0000000 0.0000000 0.6923077 1.0000000 0.0000000 0.5588235
#>   r  0.0000000 1.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
#>
#>     volkswagen
#>   4  0.0000000
#>   f  1.0000000
#>   r  0.0000000
## margin = 1 sum across row, 2 across col
```

This code calculates the column-wise proportion (as percentages) for each combination of `drv` and `manufacturer`. The

prop.table() function is used to compute the proportions. The `margin = 2` argument indicates that the proportions are to be computed across columns (`margin = 1` would compute them across rows).

tableone package

> **Tip**
>
> CreateTableOne function from tableone package could be a very useful function to see the summary table. Type `?tableone::CreateTableOne` to see for more details.

This section introduces the tableone package, which offers the `CreateTableOne` function. This function helps in creating "Table 1" type summary tables, commonly used in epidemiological studies.

```r
require(tableone)
#> Loading required package: tableone
CreateTableOne(vars = c("cyl", "drv", "hwy", "cty"), data = mpg,
               strata = "trans", includeNA = TRUE, test = FALSE)
#>                    Stratified by trans
#>                     auto(av)       auto(l3)       auto(l4)      auto(l5)
#>   n                     5              2             83            39
#>   cyl (mean (SD))  5.20 (1.10)    4.00 (0.00)    6.14 (1.62)   6.56 (1.45)
#>   drv (%)
#>      4                  0 (  0.0)      0 (  0.0)     34 (41.0)     29 (74.4)
#>      f                  5 (100.0)      2 (100.0)     37 (44.6)      8 (20.5)
#>      r                  0 (  0.0)      0 (  0.0)     12 (14.5)      2 ( 5.1)
#>   hwy (mean (SD)) 27.80 (2.59)   27.00 (4.24)   21.96 (5.64)  20.72 (6.04)
#>   cty (mean (SD)) 20.00 (2.00)   21.00 (4.24)   15.94 (3.98)  14.72 (3.49)
#>                    Stratified by trans
#>                     auto(l6)      auto(s4)      auto(s5)      auto(s6)
#>   n                     6             3             3            16
#>   cyl (mean (SD))  7.33 (1.03)   5.33 (2.31)   6.00 (2.00)   6.00 (1.59)
#>   drv (%)
#>      4                  2 (33.3)      2 (66.7)      1 (33.3)      7 (43.8)
#>      f                  2 (33.3)      1 (33.3)      2 (66.7)      8 (50.0)
#>      r                  2 (33.3)      0 ( 0.0)      0 ( 0.0)      1 ( 6.2)
```

```
#>   hwy (mean (SD)) 20.00 (2.37)  25.67 (1.15)  25.33 (6.66)  25.19 (3.99)
#>   cty (mean (SD)) 13.67 (1.86)  18.67 (2.31)  17.33 (5.03)  17.38 (3.22)
#>                   Stratified by trans
#>                    manual(m5)    manual(m6)
#>   n                   58            19
#>   cyl (mean (SD))  5.00 (1.30)   6.00 (1.76)
#>   drv (%)
#>      4               21 (36.2)      7 (36.8)
#>      f               33 (56.9)      8 (42.1)
#>      r                4 ( 6.9)      4 (21.1)
#>   hwy (mean (SD)) 26.29 (5.99)  24.21 (5.75)
#>   cty (mean (SD)) 19.26 (4.56)  16.89 (3.83)
```

The CreateTableOne function is used to create a summary table for the variables `cyl, drv, hwy, and cty` from the `mpg` dataset. The `strata = trans` argument means that the summary is stratified by the trans variable. The `includeNA = TRUE` argument means that missing values (NAs) are included in the summary. The `test = FALSE` argument indicates that no statistical tests should be applied to the data (often tests are used to compare groups in the table).

table1 package

This section introduces another package, `table1`, which can also be used to create "Table 1" type summary tables.

```r
require(table1)
#> Loading required package: table1
#>
#> Attaching package: 'table1'
#> The following objects are masked from 'package:base':
#>
#>     units, units<-
table1(~ cyl + drv + hwy + cty | trans, data=mpg)
```

	auto(av)	auto(l3)	auto(l4)	auto(l5)	auto(l6)
	(N=5)	(N=2)	(N=83)	(N=39)	(N=6)
cyl					
Mean (SD)	5.20 (1.10)	4.00 (0)	6.14 (1.62)	6.56 (1.45)	7.33 (1.03)
Median [Min, Max]	6.00 [4.00, 6.00]	4.00 [4.00, 4.00]	6.00 [4.00, 8.00]	6.00 [4.00, 8.00]	8.00 [6.00, 8.00]
drv					
f	5 (100%)	2 (100%)	37 (44.6%)	8 (20.5%)	2 (33.3%)
4	0 (0%)	0 (0%)	34 (41.0%)	29 (74.4%)	2 (33.3%)
r	0 (0%)	0 (0%)	12 (14.5%)	2 (5.1%)	2 (33.3%)
hwy					
Mean (SD)	27.8 (2.59)	27.0 (4.24)	22.0 (5.64)	20.7 (6.04)	20.0 (2.37)
Median [Min, Max]	27.0 [25.0, 31.0]	27.0 [24.0, 30.0]	22.0 [14.0, 41.0]	19.0 [12.0, 36.0]	19.0 [18.0, 23.0]
cty					
Mean (SD)	20.0 (2.00)	21.0 (4.24)	15.9 (3.98)	14.7 (3.49)	13.7 (1.86)
Median [Min, Max]	19.0 [18.0, 23.0]	21.0 [18.0, 24.0]	16.0 [11.0, 29.0]	14.0 [9.00, 25.0]	13.0 [12.0, 16.0]

The `table1()` function is used to generate a summary table for the specified variables. The formula-like syntax `(~ cyl + drv + hwy + cty | trans)` indicates that the summary should be stratified by the `trans` variable.

R Markdown

> **!** Important
>
> **Exercises in RMD**:
> The exercises for this tutorial come in R Markdown format and are provided as zip files in the relevant chapters (W, A, D, M, S). This provides you the advantage of seamlessly integrating code, outputs, and narrative text into a single document. You can then easily "knit" (the process of converting your R Markdown file into a finalized document) this file into various formats like PDF or Word (DOC / DOCX), which are widely used for academic or professional submissions. This makes it simpler for you to submit your exercises in a neat, organized manner, without the hassle of copying code or outputs from one platform and pasting them into another.

Introduction

Welcome to this tutorial on working with RMD files in RStudio! RStudio is an **IDE** that makes R programming easier and more efficient, while R Markdown (RMD) is a file format that enables you to create dynamic reports with R code and narrative text. Using R Markdown within RStudio allows you to compile your analyses and reports into a single, easily shareable document in multiple formats like HTML, PDF, or Word.

RStudio is an Integrated Development Environment (IDE), which is a software application that provides comprehensive facilities for software development. An IDE typically includes a text editor, tools for building and running code, and debugging utilities.

Prerequisites

Before starting this tutorial, make sure you have both R and RStudio installed on your computer, as explaied in an earlier tutorial.

Knitting RMD

This document shows how to work with an RMD file. We can create dynamic documents, e.g., a document with simple plain text combined with R code and its outputs. Note that RMD files are designed to be used with the R package `rmarkdown`. In RStudio IDE, the `rmarkdown` package could be already installed.

```
# install.packages("rmarkdown")
```

Open on RStudio

Let us open an RMD file in RStudio. From the file menu (on the side), we can create a new RMD document. First, we need to click on the + symbol to create a new file, as follows:

Second, we need to click `R markdown...` to create a new RMD document as follows:

We will see a pop-up window as follows:

We can select whether we want to convert our RMD file to an `HTML`, `PDF`, or a `Word` document. These options can also be selected later. Let us select the default option (HTML) and press OK. We will see the markdown file as shown in the picture below:

RStudio
File Edit Code View Plots Session Buil
Go to
R Script Ctrl+Shift+N
Quarto Document...
Quarto Presentation...
R Notebook
R Markdown...
Shiny Web App...
Plumber API...
Text File
on Save

New R Markdown
Document
Presentation
Shiny
From Template
Title:
Untitled
Author:
Date:
2023-09-20
Use current date when rendering document
Default Output Format:
HTML
Recommended format for authoring (you can switch to PDF or Word output anytime).
PDF
PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).
Word
Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).
Create Empty Document
OK
Cancel

Knit

We use the `knit` option to create a document (e.g., making a PDF, HTML, or Word document) from the RMD file. Before knitting, we need to save the file. Let us save the file as **working.RMD**. After saving the file, we can knit it by clicking on the `knit` option, as shown below:

The term "knit" may sound a bit strange in the context of programming. However, it aptly describes the process of combining your R code and narrative text to produce a cohesive, final document. Think of it as "weaving" your code and text together into various output formats like HTML, PDF, or Word.

Once we knit the file, it will produce an `HTML` output, since our default option was HTML.

For formats other than `HTML` (e.g., `PDF` or `Word`), we can click

on the dropdown menu:

Let us select `Knit to Word` and knit it. Once the file is rendered, RStudio will show us a preview of the output in a word file and save the file in our **working directory**. We can also see that `Word` is added as another output:

When you're working in RStudio, all your files and outputs will be saved in a 'working directory.' This is simply the folder on your computer where RStudio will look for files and save outputs. To find out what your current working directory is, you can run the command `getwd()` in the R console.

In the R terminal, we can see that a `Word` document is created, which is stored in our working directory:

Similarly, we can create a pdf by clicking `Knit to PDF` option from the `Knit` menu. However, we could see an error message as follows:

```
processing file: working.Rmd

"C:/Program Files/RStudio/resources/app/bin/quarto/bin/tools/pandoc" +RTS -K512m -RTS working.knit.md --to latex --fro
m markdown+autolink_bare_uris+tex_math_single_backslash --output working.tex --lua-filter "C:\Users\Belal\AppData\Loca
l\R\win-library\4.3\rmarkdown\rmarkdown\lua\pagebreak.lua" --lua-filter "C:\Users\Belal\AppData\Local\R\win-library\4.
3\rmarkdown\rmarkdown\lua\latex-div.lua" --embed-resources --standalone --highlight-style tango --pdf-engine pdflatex
--variable graphics --variable "geometry:margin=1in"
output file: working.knit.md

Error in loadNamespace(x) : there is no package called 'tinytex'
Calls: <Anonymous> ... loadNamespace -> withRestarts -> withOneRestart -> doWithOneRestart
Execution halted

No LaTeX installation detected (LaTeX is required to create PDF output). You should install a LaTeX distribution for y
our platform: https://www.latex-project.org/get/

  If you are not sure, you may install TinyTeX in R: tinytex::install_tinytex()

  Otherwise consider MiKTeX on Windows - http://miktex.org

  MacTeX on macOS - https://tug.org/mactex/
  (NOTE: Download with Safari rather than Chrome _strongly_ recommended)

  Linux: Use system package manager
```

It is important to note that RStudio does not build PDF documents from scratch. If we want to create PDF documents using RMD, we must have a LaTeX distribution installed on our computer. There are several options for LaTeX distributions, including MiKTeX, MacTeX, TeX Live, and so on. However, the recommended option for R Markdown users is TinyTeX. We can install TinyTeX using the R package `tinytex`. To install the package, run the following command: `install.packages("tinytex")`.

```
72:387   Open an RMD file on RStudio
Console   Terminal   Background Jobs
R 4.3.1 · ~/GitHub/OER-Advanced-Epidemiological-Methods/
>
> install.packages("tinytex")
```

LaTeX is a typesetting system commonly used for technical and scientific documentation. It is required for converting R Markdown documents to PDF format because it provides the text formatting commands that the `rmarkdown` package uses behind the scenes.

Once the `tinytex` package installation is complete, we can type `tinytex::install_tinytex()` to install the LaTeX distribution on our computer.

```
76:4   Open an RMD file on RStudio
Console   Terminal   Background Jobs
R 4.3.1 · ~/GitHub/OER-Advanced-Epidemiological-Methods/
>
> tinytex::install_tinytex()
```

TinyTeX is a large package (~123 MB). The installation time will vary depending on your machine. Once the installation is complete, we can click `Knit to PDF`. Similar to the Word file, RStudio will display a preview of the PDF output and save the

PDF in our working directory. We will also see that a PDF file has been created:

```
1/21   R Markdown
Console   Terminal ×   Render ×   Background Jobs ×
.../New folder/working.Rmd

processing file: working.Rmd

"C:/Program Files/RStudio/resources/app/bin/quarto/bin/tools/pandoc" +RTS -K512m -R1
m markdown+autolink_bare_uris+tex_math_single_backslash --output working.tex --lua-f
l\R\win-library\4.3\rmarkdown\rmarkdown\lua\pagebreak.lua" --lua-filter "C:\Users\Be
3\rmarkdown\rmarkdown\lua\latex-div.lua" --embed-resources --standalone --highlight-
--variable graphics --variable "geometry:margin=1in"
output file: working.knit.md

Output created: working.pdf
```

Working with RMD

Now we are ready to start writing plain text intermixed with embedded R code. For plain text, we can use the whitespace:

```
## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word
documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the
output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```{r cars}
summary(cars)
```
```

On the other hand, to embed a chunk of R code into our report, we use R code chunks. An R chunk surrounds the code with two lines that each contain three backticks. After the first set of backticks, we include `{r}`, which alerts `knitr` that we are going to include a chunk of R code:

```
## R Markdown

This is an R Markdown document. Markdown is a simple fc
documents. For more details on using R Markdown see <ht

When you click the **Knit** button a document will be g
output of any embedded R code chunks within the documer

```{r cars}
summary(cars)
```
```

Code chunks are segments of code that are contained within an R Markdown document. They allow you to run R code within the document itself, making your report dynamic and reproducible.

Below are some codes:

We can knit the file to see the document, which will include plain text, R code, and outputs from the R code. We can also

```
```{r cars}
data(cars)
summary(cars)
```
```

see the output from a code chunk without knitting the entire file. For example, we can click the arrow on the right-hand side to execute the current code chunk:

Now we can see the following outputs:

Please also explore the drop down menu under `Run` to see the further options, including run the current code chunk, run all code chunk above, etc.

To omit the code from the final report while still including the results, add the argument `echo = FALSE`. This will place a copy of the results into your report.

In the final report (e.g., Word or PDF), we often want to omit the code and only show the outputs. To do this, we added the argument `echo = FALSE` in the R code chunk:

The resulting output will look as follows:

Besides `echo = FALSE`, there are several other options you can include in your code chunks to control their behavior, like `eval = FALSE` if you don't want to evaluate the code, or `message = FALSE` to hide messages. Take a look at the author's page of comprehensive list of chunk options.

Tips and Troubleshooting

- If the `knit` button is grayed out, make sure you have saved your RMD file first.
- Encountering LaTeX errors? Make sure you've installed a LaTeX distribution like TinyTeX.

```{r pressure, echo=FALSE}
plot(pressure)
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
data(cars)
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00  
```

Including Plots

You can also embed plots, for example:

Tip

The following links could also be useful if you want to learn more:

- R Markdown Cheat Sheet
- Introduction to R Markdown
- R Markdown: The Definitive Guide
- Reports with R Markdown

Video content (optional)

Tip

For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

R Functions (W)

> **Note**
>
> This review/summary page provides an extensive list of R functions tailored for data wrangling tasks that we have used in this chapter. Each function is systematically described, highlighting its primary package source and its specific utility.

To learn more about these functions, readers can:

1. **Use R's Built-in Help System**: For each function, access its documentation by prefixing the function name with a question mark in the R console, e.g., `?as.factor`. This displays the function's manual page with descriptions, usage, and examples.

2. **Search Websites**: Simply Google, or visit the CRAN website to search for specific function documentation. Websites like Stack Overflow and RStudio Community often have discussions related to R functions.

3. **Tutorials and Online Courses**: Platforms like DataCamp, Coursera, and edX offer R courses that cover many functions in depth. Also there are examples of dedicated R tutorial websites that you might find useful. One example is "Introduction to R for health data analysis" by Ehsan Karim, An Hoang and Qu.

4. **Books**: There are numerous R programming books, such as "R for Data Science" by Hadley Wickham and "The Art of R Programming" by Norman Matloff.

5. **Workshops and Webinars**: Institutions and organizations occasionally offer R programming workshops or webinars.

Whenever in doubt, exploring existing resources can be highly beneficial.

Function_name	Package_name	Use
as.factor	base	Converts a variable to factors. ‘as.factor‘ is a wrapper for the ‘factor‘ functio
cbind	base	Merges matrices.
CreateTableOne	tableone	Creates a frequency table.
data.frame	base	Creates a dataset with both numeric and character variables. Requires uniqu
dim	base	Returns the dimensions of a data frame (rows x columns).
filter	dplyr	Subsets a dataset by selecting a sub-population.
function	base	Used to define custom functions, e.g., for calculating standard deviation.
head	base	Displays the first six elements of an object (e.g., a dataset). ‘tail‘ displays th
is.na	base	Checks for missing values in a variable.
levels	base	Displays the levels of a factor variable.
list	base	Stores vectors, matrices, or lists of differing types.
mode	base	Determines the type of a variable.
na.omit	base/stats	Removes all rows with missing values from a dataset.
names	base	Displays names of objects, e.g., variable names of a data frame.
nlevels	base	Shows the number of levels in a factor variable.
nrow	base	Returns the dimensions of a data frame. ‘nrow‘ gives row count and ‘ncol‘ gi
plot	base/graphics	Draws scatter plots or line graphs.
print	base	Prints the output to console.
prop.table	base	Displays percentage summary for a table.
rbind	base	Appends matrices row-wise.
read.csv	base/utils	Reads data from a CSV file.
relevel	base/stats	Changes the reference group of a factor variable.
sasxport.get	Hmisc	Loads data in the SAS format.
save	base	Saves R objects, such as datasets.
select	dplyr	Selects specified variables from a dataset.
set.seed	base	Sets a seed for random number generation ensuring reproducibility.
str	base/utils	Displays the structure of a dataset, including data type of variables.
subset	base, dplyr	Subsets a dataset by selecting a sub-population.
summary	base	Provides a summary of an object, like variable statistics.
table	base	Displays frequency counts for a variable.
write.csv	base/utils	Saves a data frame to a CSV file in a specified directory.

Quiz (W)

> ⚠ Warning
>
> We are currently in the beta testing phase for our quiz page updates, so please be aware that some responses may not be accurate. Thank you for your patience.
> For those interested in downloading the quiz questions, you'll find a section below dedicated to downloading.

Live Quiz

You can click on the live quiz options to either see hints of why your response may be incorrect (response marked in red), or whether your choice of response is correct (response marked in green):

Download Quiz

1. **Downloading the File:**
    - Navigate to the link: See here. If not downloading immediately, right-click on the link and select "Save link as…" from the dropdown menu.
    - Alternatively click here for downloading the quizzes.
    - Choose a destination folder on your computer where you'd like to save the file (e.g., Desktop). **Remember this location**, as you'll need to navigate to it later.
2. **Setting Up RStudio:**

- If you don't have RStudio installed, see the download link in here.
- Launch RStudio after installing.

3. **Installing Necessary R Packages:**

   - Before running the R Markdown file, ensure you have the required packages. In RStudio's console (located at the bottom left by default), enter the following commands to install them:

```
install.packages("learnr")
install.packages("xfun")
```

4. **Opening and Running the File:**

   - In RStudio, go to `File` > `Open File...`.
   - Navigate to the folder where you saved the `Rmd` file and select it to open.
   - Once the file is open in RStudio, you'll see a "Run Document" button (green) at the top of the script editor. Click on it to run the R Markdown Quiz file.

App (W)

Below is an example of an app that utilizes the `mpg` dataset from the `ggplot2` package following the tutorial materials. Users can subset the data and generate a stratified Table 1 from it.

> Tip
>
> **Web App may need some time to load the necessary files, depending on your internet speed.**

> Important
>
> If you want to run the app locally in your computer instead for faster load time: you can download the app directly in your R via `shiny::runGitHub("EpiShinyLiveW", "ehsanx", subdir = "app")`. Make sure that `shiny`, `tableone` and `ggplot2` packages are installed in your computer/R.

> Warning
>
> **Bug Report**:
> Fill out this form to report any issues with the app.

Exercise (W)

Tip

You cannot input any codes on this website but need to download the exercise files on your computer. You can download all of the related files in a zip file **wranglingEx.zip** from Github folder folder, or just by clicking this link directly.

- Navigate to the GitHub folder (above link) where the ZIP file is located.
- Click on the file name (above zip file) to open its preview window.
- Click on the Download button to download the file. If you can't see the Download button, click on "Download Raw File" link that should appear on the page.

Important

Once you download and unzip the files, you should see the following files:

Name	Type	Compressed size	Password ...	Size
Data	File folder			
Images	File folder			
wranglingE	Adobe Acrobat Document	244 KB	No	
wranglingE	RMD File	4 KB	No	

- You will work with the file `wranglingE.RMD`

> **Important**
>
> Before you start:
>
> - You need R and RStudio to complete this exercise. See the R basics chapter on how to download and install R and RStudio on your computer.
> - Please knit your file at least once and check whether a pdf file is created. See the R Markdown chapter if you need help regarding working with an RMD file.

Problem Statement

Use the functions we learned in Lab 1 to complete Lab 1 Exercise. We will use Right Heart Catheterization Dataset saved in the folder named 'Data/wrangling/'. The variable list and description can be accessed from Vanderbilt Biostatistics website.

You can access the original table from this paper (doi: 10.1001/jama.1996.03540110043030). We have modified the table and corrected some issues. Please knit your file once you finished and submit the knitted file **ONLY**.

```r
# Load required packages
library(dplyr)
library(tableone)
```

```r
# Data import: name it rhc
#rhc <- ...("Data/wrangling/rhc.csv", ...)
```

Problem 1: Basic Manipulation [60%]

(I) Continuous to Categories: Change the Age variable into categories below 50, 50 to below 60, 60 to below 70, 70 to

below 80, 80 and above [Hint: the `cut` function could be helpful]

(II) Re-order: Re-order the levels of race to white, black and other

(III) Set reference: Change the reference category for gender to Male

(IV) Count levels: Check how many levels does the variable "cat1" (Primary disease category) have? Regroup the levels for disease categories to "ARF","CHF","MOSF","Other". [Hint: the `nlevels` and `list` functions could be helpful]

(V) Rename levels: Rename the levels of "ca" (Cancer) to "Metastatic","None" and "Localized (Yes)", then re-order the levels to "None","Localized (Yes)" and "Metastatic"

(VI) comorbidities:

- Create a new variable called "numcom" to count number of comorbidities illness for each person (12 categories) [Hint: the `rowSums` command could be helpful],
- Report maximum and minimum values of numcom:

```
# See head of comorbidities
# head(rhc[,c("cardiohx", "chfhx", "dementhx", "psychhx", "chrpulhx", "renalhx",
#             "liverhx", "gibledhx", "malighx", "immunhx", "transhx", "amihx")])

# your codes here
```

(VII) Anlaytic data: Create a dataset that has only the following variables

- age, sex, race, cat1, ca, dnr1, aps1, surv2md1, numcom, adld3p, das2d3pc, temp1, hrt1, meanbp1, resp1, wblc1, pafi1, paco21, ph1, crea1, alb1, scoma1, swang1
- name the dataset as rhc2

Problem 2: Table 1 [10%]

Re-produce the sample table 1 from the rhc2 data (see the Table below). In your table, the variables should be ordered as the same as the sample. Please re-level or re-order the levels if needed. [Hint: the `tableone` package might be useful]

	No RHC	RHC
n	3551	2184
age (%)		
[-Inf,50)	884 (24.9)	540 (24.7)
[50,60)	546 (15.4)	371 (17.0)
[60,70)	812 (22.9)	577 (26.4)
[70,80)	809 (22.8)	529 (24.2)
[80, Inf)	500 (14.1)	167 (7.6)
sex = Female (%)	1637 (46.1)	906 (41.5)
race (%)		
white	2753 (77.5)	1707 (78.2)
black	585 (16.5)	335 (15.3)
other	213 (6.0)	142 (6.5)
cat1 (%)		
ARF	1581 (44.5)	909 (41.6)
CHF	247 (7.0)	209 (9.6)
Other	955 (26.9)	208 (9.5)
MOSF	768 (21.6)	858 (39.3)
ca (%)		
None	2652 (74.7)	1727 (79.1)
Localized (Yes)	638 (18.0)	334 (15.3)
Metastatic	261 (7.4)	123 (5.6)
dnr1 = Yes (%)	499 (14.1)	155 (7.1)
aps1 (mean (SD))	50.93 (18.81)	60.74 (20.27)
surv2md1 (mean (SD))	0.61 (0.19)	0.57 (0.20)
numcom (mean (SD))	1.52 (1.17)	1.48 (1.13)
adld3p (mean (SD))	1.24 (1.86)	1.02 (1.69)
das2d3pc (mean (SD))	20.37 (5.48)	20.70 (5.03)
temp1 (mean (SD))	37.63 (1.74)	37.59 (1.83)
hrt1 (mean (SD))	112.87 (40.94)	118.93 (41.47)
meanbp1 (mean (SD))	84.87 (38.87)	68.20 (34.24)
resp1 (mean (SD))	28.98 (13.95)	26.65 (14.17)
wblc1 (mean (SD))	15.26 (11.41)	16.27 (12.55)

	No RHC	RHC
pafi1 (mean (SD))	240.63 (116.66)	192.43 (105.54)
paco21 (mean (SD))	39.95 (14.24)	36.79 (10.97)
ph1 (mean (SD))	7.39 (0.11)	7.38 (0.11)
crea1 (mean (SD))	1.92 (2.03)	2.47 (2.05)
alb1 (mean (SD))	3.16 (0.67)	2.98 (0.93)
scoma1 (mean (SD))	22.25 (31.37)	18.97 (28.26)

Problem 3: Table 1 for subset [10%]

Produce a similar table as Problem 2 but with only male sex and ARF primary disease category (cat1). Add the overall column in the same table. [Hint: `filter` command could be useful]

Problem 4: Considering eligibility criteria [20%]

Produce a similar table as Problem 2 but only for the subjects who meet all of the following eligibility criteria: (i) age is equal to or above 50, (ii) age is below 80 (iii) Glasgow Coma Score is below 61 and (iv) Primary disease categories are either ARF or MOSF. [Hint: `droplevels.data.frame` can be a useful function]

Optional [0%]

Optional 1: Missing values

(I) Any variables included in rhc2 data had missing values? Name that variable. [Hint: `apply` function could be helpful]

(II) Count how many NAs does that variable have?

(III) Produce a table 1 for a complete case data (no missing observations) stratified by `swang1`.

Optional 2: Calculating variance of a sample

Write a `function` for Bessel's correction to calculate an unbiased estimate of the population variance from a finite sample (a vector of 100 observations, consisting of numbers from 1 to 100).

```
Vector <- 1:100

#variance.est <- function(?){?}

#variance.est(Vector)
```

Hint: Take a closer look at the functions, loops and algorithms shown in lab materials. Use a `for loop`, utilizing the following pseudocode of the algorithm:

Naïve algorithm [edit]

A formula for calculating the variance of an entire population of size N is:

$$\sigma^2 = \overline{(x^2)} - \bar{x}^2 = \frac{\sum_{i=1}^{N} x_i^2 - (\sum_{i=1}^{N} x_i)^2/N}{N}.$$

Using Bessel's correction to calculate an unbiased estimate of the population variance from a finite sample of n observations, the formula is:

$$s^2 = \left(\frac{\sum_{i=1}^{n} x_i^2}{n} - \left(\frac{\sum_{i=1}^{n} x_i}{n}\right)^2\right) \cdot \frac{n}{n-1}.$$

Therefore, a naive algorithm to calculate the estimated variance is given by the following:

- Let $n \leftarrow 0$, Sum $\leftarrow 0$, SumSq $\leftarrow 0$
- For each datum x:
  - $n \leftarrow n + 1$
  - Sum $\leftarrow$ Sum $+ x$
  - SumSq $\leftarrow$ SumSq $+ x \times x$
- Var = (SumSq − (Sum × Sum) / n) / (n − 1)

Verify that estimated variance with the following variance function output in R:

```
var(Vector)
#> [1] 841.6667
```

Part II

Accessing data

Background

Surveys serve as a pivotal tool for collecting and evaluating health-related information on a national scale. More often than not, it's the governmental bodies that take the lead in gathering this data. Recognizing the value of this information, many governments not only compile and analyze these datasets but also ensure they are accessible to the public, especially for research purposes. In this guide, we will delve into an array of survey methodologies, provide illustrative examples, and guide you through the process of downloading pertinent data from both Canadian and American repositories. To make this more tangible, we will conclude with a hands-on example, showcasing how to replicate findings from an academic paper that leveraged one of these publicly available datasets.

> **!** Important
>
> **Datasets**:
> All of the datasets used in this tutorial can be accessed from this GitHub repository folder

Now that we are familiar with R basics and data wrangling, we are now dedicating a chapter to accessing and downloading nationally representative survey datasets is pivotal for a hands-on epidemiological tutorial. These datasets, often rich in information and reflective of diverse populations, serve as the backbone for real-world analysis. By guiding learners on how to obtain these datasets, the book ensures that they not only grasp theoretical concepts but also gain practical experience working with authentic, large-scale data. This approach bridges the gap between theory and practice, allowing readers to apply learned techniques on datasets that mirror real-world complexities, thereby enhancing the relevance and applicability of their analytical skills.

Overview of tutorials

Survey data sources

The tutorial lists primary complex survey data sources, including the Canadian Community Health Survey and National Health and Nutrition Examination Survey, with several offering dedicated R packages for data access.

Descriptions of data sources

This tutorial provides comprehensive instructions on how to import and process health survey datasets, specifically focusing on the Canadian Community Health Survey (CCHS), Na-

tional Health and Nutrition Examination Survey (NHANES), and National Health Interview Survey (NHIS).

Importing CCHS to R

The section provides detailed steps for importing the Canadian Community Health Survey dataset from the UBC library into RStudio, with processing options using SAS, the free software PSPP, and directly in R.

Importing NHANES to R

The tutorial guides users on how to access and import the NHANES dataset from the CDC website into RStudio, detailing the dataset's structure and providing methods both manually and using an R package.

Reproducing results

The tutorial guides users through accessing, processing, and analyzing NHANES data to reproduce the results from a referenced article using R code.

Importing NHIS to R

This chapter serves as a tutorial on accessing and importing the National Health Interview Survey (NHIS) dataset from the US Centers for Disease Control and Prevention (CDC) website into RStudio. The NHIS is an annual cross-sectional survey managed by the CDC, offering insights into population disease prevalence, disability extent, and utilization of health care services. The data files are available in various formats, including ASCII, CSV, and SAS. Users can combine datasets from different years; however, tracing the same individual across cycles is not feasible. The chapter provides step-by-step guidance on downloading the NHIS dataset, particularly the 'Adult' data from 2021, verifying the imported data, and merging datasets

within the same survey cycle using the unique household identifier.

Linking mortality data

This tutorial provides instructions on how to link public-use US mortality data with NHANES dataset, with the option to do the same with NHIS. The process involves downloading the mortality data in `dat` format from the CDC website, loading it into the R environment, and then merging it with the NHANES data using a unique identifier. Various variables related to mortality status, underlying causes of death, and other health-related factors are explained, and Table 1 is created with unweighted and weighted statistics.

> **Note**
>
> **What is Coming Next**:
> The subsequent chapter on Research Questions serves as a valuable guide for constructing an analytics-driven data set tailored to your specific research queries. It will cover crucial aspects such as the types of variables to collect and how to set eligibility criteria, followed by approaches to data analysis based on your research questions. It's important to note that research questions can fall into two main categories: predictive or causal. For a deeper understanding of variable selection and analytical tools suited to these types of questions, the chapters on the Roles of Variables and Predictive Models offer insightful guidance.

> **Warning**
>
> **Bug Report**:
> Fill out this form to report any issues with the tutorial.

Concepts (A)

Model-based approach

The model-based approach to statistical analysis is heavily reliant on the specification of a probability model for data generation, typically assuming that data come from an infinite population that follows a specific distribution, such as the Normal distribution. Inferences about the population, including point estimates and hypothesis testing, are made based on how well the sample data fit these model assumptions.

Design-based approach

The design-based approach emphasizes the use of sampling methods and the design of the study itself to make inferences about a real/finite population. The design-based approach takes into account the actual structure of the data collection process to make inferences, ensuring that each unit in the population has a known and often non-zero chance of being included in the sample, thus addressing the potential biases and variance issues arising from the sampling design. This approach is critical in understanding and analyzing data from surveys with complex designs, including those with stratification, clustering, and weighting.

Reading list

Key reference:

- (Steven G. Heeringa, West, and Berglund 2017) (chapter 2)

Optional reading:

- (Lumley 2011) (chapter 1)
- (Vittinghoff et al. 2011b) (chapter 12)
- (Bilder and Loughin 2014) (section 6.3)

Video Lessons

Model-based approach

Review materials from pre-requisite statistics courses (optional)

Design-based approach

What is included in this Video Lesson:

- Model-based approach review: 0:00
- Design-based approach: 1:15
- Types of sampling techniques 6:46
- Statistical inference 8:25
- NHANES 12:02
- Survey weight 20:40
- CCHS download 23:45
- NHANES download 24:50
- NHANES sampling design 27:24
- How to find NHANES data from CDC website 27:42

The timestamps are also included in the YouTube video description.

Video Lesson Slides

Links

- Google Slides
- PDF Slides

- Model-based approach (Review/optional content)
- Design-based approach

References

Survey data sources

The tutorial discusses complex survey data and highlights potential data sources. Key datasets with survey features include the Canadian Community Health Survey (CCHS), the National Health and Nutrition Examination Survey (NHANES), and the European Social Survey (ESS), among others. Many of these sources, like NHANES and ESS, have specific R packages for data retrieval. In addition, there are other data sources such as the Vanderbilt Biostatistics Datasets and the World Bank Open Data, with the latter also offering dedicated R packages for data access.

Survey features

Generally survey features include - Strata - Cluster/primary sampling unit (PSU) - Weight, e.g., interview weight, survey weight

Dataset with survey features

- Canadian Community Health Survey - Annual Component CCHS
  - Download link UBC library
- National Health and Nutrition Examination Survey NHANES
  - R packages to download data: nhanesA, RNHANES
- National Longitudinal Study of Adolescent to Adult Health [Add Health], 1994-2008 ICPSR 21600
- European Social Survey ESS
  - R package to download data: essurvey

- Behavioral Risk Factor Surveillance System BRFSS
- Bureau of Economic Analysis BEA
- US National Vital Statistics System NVSS
- Demographic and Health Surveys DHS

Others

- Vanderbilt Biostatistics Datasets link
- World Bank Open Data WBOD
  - R packages to download data: wbstats, WDI

Descriptions

This tutorial introduces CCHS as a cross-sectional survey that collects health-related data and discusses its objectives and data usage. Additionally, it highlights the survey's evolution and redesigns. For NHANES, the tutorial covers the importance of the dataset, its sampling procedures, history, data files, and documents. It also discusses how to combine data from different cycles, handle missing data, and deal with outliers. NHIS, another CDC-supported survey, is briefly introduced as a source of annual health-related data. There are some changes in the dataset due to the COVID-19 pandemic, e.g., fewer variables compared to pre-pandemic datasets. Otherwise, the main purpose of these datasets remained the same across survey cycles.

CCHS

Overview

CCHS is a cross-sectional survey that collects vital health-related data, including health status, healthcare utilization, and health determinants, from the Canadian population. Available in both official languages, this survey relies on a substantial sample size to provide reliable estimates at various geographical levels every two years.

Objectives of the CCHS

The CCHS has four primary objectives: supporting health surveillance programs at national, provincial, and intra-provincial levels; offering a single data source for health research on small populations and rare characteristics; providing timely and easily accessible information to a diverse user

community; and maintaining flexibility to address emerging health issues within the population.

Data Products and Usage

The CCHS generates annual microdata files and combines two years of data for analysis. Users can also combine data from different years to study specific populations or rare characteristics. The data is primarily used for health surveillance and population health research, benefiting federal and provincial health departments, social service agencies, government bodies, and researchers from various fields. Non-profit health organizations and the media also utilize CCHS results to raise awareness about health concerns.

Evolution and Redesigns

The CCHS started collecting data in 2001, transitioning to annual data collection in 2007 with a sample size adjustment to 65,000 respondents per year. It has undergone two significant redesigns to enhance its utility. The 2015 redesign updated sampling methods, adopted a new sample frame, modernized health content, and reviewed the target population. In 2022, the survey underwent another redesign, further updating content and transitioning to an online electronic questionnaire (EQ) for direct self-reporting by selected respondents. Both redesigns involved extensive consultations with stakeholders, including federal, provincial, and territorial partners, health region authorities, and academics.

NHANES

This section covers

1. Introduction to the NHANES dataset, highlighting its significance in evaluating the health and nutritional status of U.S. adults and children.

2. Sampling Procedure details, explaining the multi-stage sampling strategy and emphasizing the importance of using survey features like weights, strata, and primary sampling units for population-level estimates.
3. Survey History with a visualization representing different NHANES survey cycles.
4. NHANES Data Files and Documents:

- Explains the data's file format, mostly in SAS transport file format (.xpt).
- Breaks down the NHANES components, which include demographics, dietary, examination, laboratory, and questionnaire data.
- Provides guidelines on combining data from different cycles and handling missing data or outliers.

Overview

National Center for Health Statistics (NCHS) conducts National Health and Nutrition Examination Survey (NHANES) (CDC,NCHS 2023). These surveys are designed to evaluate the health and nutritional status of U.S. adults and children. These surveys are being administered in two-year cycles or intervals starting from 1999-2000. Prior to 1999, a number of surveys were conducted (e.g., NHANES III), but in our discussion, we will mostly restrict our discussions to continuous NHANES (e.g., NHANES 1999-2000 to NHANES 2017-2018).

CDC,NCHS (2023)

Sampling Procedure:

It is a probabilistic sample (we know probability of getting selected for all individuals). This sample is unlikely to be representative of the entire population, as some under/oversampling occurs (unlike SRS), and samples may be dependent (due to proximity of some samples). For example, household with the following characteristics may be oversampled in NHANES, e.g., African Americans, Mexican Americans, Low income White Americans, Persons age 60+ years.

Sampling Procedure:

- not obtained via simple random sample
- multistage sample designs
- A sample weight is assigned to each sample person where weight = the number of people in the target population represented by that sample person in NHANES

NHANES used multistage sample designs:

- Stage 1: PSU/clusters = geographically contiguous counties. 50 states - divided into ~3100 counties. Each PSU is assigned to a strata (e.g., urban/rural or PSU size etc.). The counties are randomly/PPS selected using a 2-per-stratum design. Complex sample variance estimation requires PSU + strata (masking involved).
- Stage 2: each selected county is broken into segments (with at least ~50-100 housing units). Segments are randomly/PPS selected.
- Stage 3: each selected segment is divided into households. Households are randomly selected.
- Stage 4: Within each sampled household, an individual is randomly selected.

To obtain population-level estimate, we must utilize the survey features (weights, strata, PSU/cluster)

Survey history

Overall NHANES survey history

NHANES datafile and documents

0.0.0.0.1 * File format

The Continuous NHANES files are stored in the NHANES website as SAS transport file formats (.xpt). You can import this data in any statistical package that supports this file format.

0.0.0.0.2 * Continuous NHANES Components

Continuous NHANES components separated to reduce the amount of time to download and documentation size:

Broadly, continuous NHANES data are available in 5 categories:

- Demographics
- Dietary
- Examination
- Laboratory
- Questionnaire

0.0.0.0.3 * Combining data

0.0.0.0.3.1 * Different cycles

It is possible to combine datasets from different years/cycles together in NHANES. However, NHANES is a cross-sectional data, and identification of the same person accross different cycles is not possible in the public release datasets. For appending

data from different cycles, please make sure that the variable names/labels are the same/identical in years under consideration (in some years, names and labels do change).

0.0.0.0.3.2 * Within the same cycle

Within NHANES datasets in a given cycle, each sampled person has an unique identifier sequence number (variable `SEQN`).

The following data have not been released on the NHANES website as public release files due to confidentiality concerns:

- adolescent data on alcohol use
- smoking
- sexual behavior
- reproductive health and drug use

0.0.0.0.4 * Missing data and outliers

CDC (2023) recommends:

Key points on NHANES data analysis and missing data handling:

CDC (2023)

1. If less than 10% of your data for a variable are missing, it's generally acceptable to proceed with your analysis without further evaluation or adjustment. However, when more than 10% of data is missing, assess if the missing values are evenly distributed across socio-demographic characteristics. Consider options like imputation or adjusted weights if necessary.
2. Identify and treat 'refusal' or 'do not know' responses as missing data to prevent distorted results in statistical analyses. Recode these responses as missing values, using either a period (.) for numeric variables or a blank for character variables.
3. Be cautious about outliers with exceptionally large weights, as they can significantly impact your estimates. Analysts should decide whether to include or exclude these influential outliers from the analysis, taking into account their potential impact on results.

0.0.0.0.5 * NHANES documents

NHIS

Like NHANES, National Health Interview Survey (NHIS) is supported by the CDC and is a large-scale multi-stage cross-sectional survey. The NHIS survey includes information on population disease prevalence, extent of disability, and use of health care services. In contrast to the NHANES that provides data every 2 years, NHIS provides data annually.

The following websites could be helpful: - For more information about NHANES design.

- Visit US CDC website and do a variable keyword search based on your research interest (e.g., arthritis).

To obtain population-level estimate, we must utilize the survey features (weights, strata, PSU/cluster)

References

Importing CCHS to R

Overview

This section provides comprehensive instructions on how to import the Canadian Community Health Survey (CCHS) dataset from the UBC library site to the RStudio environment. The process starts with downloading the CCHS data from the UBC library site and includes step-by-step visual guides for each stage. Three primary options are provided to process and format the data:

1. Using the commercial software SAS.
2. Utilizing the free software PSPP, an alternative to SPSS.
3. Directly processing the data in R.

For each option, users are guided on how to download, install, access, read, save, and check the dataset. The objective is to help users acquire, visualize, and manipulate the CCHS dataset seamlessly using various software applications.

Downloading CCHS data from UBC

- **Step 1**: Go to dvn.library.ubc.ca, and press 'log-in'

- **Step 2**: Select 'UBC' from the dropdown menu

- **Step 3**: Enter your CWL or UBC library authentication information

- **Step 4**: Once you log-in, search the term 'cchs' in the search-box

- **Step 5**: For illustrative purposes, let us work with the Cycle 3.1 of the CCHS dataset from the list of results. In that case, type 'cchs 3.1'

- **Step 6**: CCHS Cycle 3.1 information

- **Step 7**: Choose the 'Data: CD' from the menu

- **Step 8**: Download the entire data (about 159 MB) as a zip file

- **Step 9**: Accept the 'terms of use'

- **Step 10**: Select a directory to download the zip file. The path of the download directory is important (we need to use this `path` exactly later). For example, below we are

in "C:\CCHS\" folder, but we will create a “Data” folder there, so that the download path is "C:\CCHS\Data\".

- **Step 11**: Extract the zip file

- **Step 12**: Be patient with the extraction

- **Step 13**: Once extraction is complete, take a look at the folders inside. You will see that there is a folder named 'SAS_SPSS'

Reading and Formatting the data

Option 1: Processing data using SAS

SAS is a commercial software. You may be able to get access to educational version. In case you don't have access to it, later we outline how to use free packages to read these datasets.

- **Step 1**: Inside that 'SAS_SPSS' folder, find the file *hs_pfe.sas*. It is a long file, but we are going to work on part of it. First thing we want to do it to change all the directory names to where you have unzipped the downloaded file (for example, here the zip file was extracted to C:/CCHS/Data/cchs_cycle3-1CD/). We only need the first part of the code (as shown below; only related to data 'hs'). Delete the rest of the codes for now. The resulting code should like like this:

```
%include "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hs_pfe.sas";

data hs;
        %let datafid="C:\CCHS\Data\cchs_cycle3-1CD\Data\hs.txt";
        %include "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hs_i.sas";
        %include "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hs_fmt.sas";
        %include "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hs_lbe.sas";
run;
```

Once the modifications are done, submit the codes in SAS. Note that, the name of the data is 'hs'.

- **Step 2**: Once you submit the code, you can check the log window in SAS to see how the code submission went. It should tell you how many observations and variables were read.

- **Step 3**: If you want to view the dataset, you can go to 'Explorer' window within SAS.

- **Step 4**: Generally, if you haven't specified where to load the files, SAS will by default save the data into a library called 'Work'

- **Step 5**: Open that folder, and you will be able to find the dataset 'Hs'.

- **Step 6**: Right click on the data, and click 'open' to view the datafile.

- **Step 7**: To export the data into a CSV format data (so that we can read this data into other software packages), ckick 'Menu'.

- **Step 8**: then press 'Export Data'.

- **Step 9**: choose the library and the data.

- **Step 10**: choose the format in which you may want to save the existing data.

- **Step 11**: also specify where you want to save the csv file and the name of that file (e.g., cchs3.csv).

- **Step 12**: go to that directory to see the file cchs3.csv

- **Step 13**: If you want to save the file in SAS format, you can do so by writing the following sas code into the 'Editor' window. Here we are saving the data Hs within the Work library in to a data called cchs3 within the SASLib library. Note that, the directory name has to be where you want to save the output file.

```
LIBNAME SASLib "C:\CCHS\Data";
DATA SASLib.cchs3;
    set Work.Hs;
run;
```

Submit these codes into SAS:

- **Step 13**: go to that directory to see the file cchs3.sas7dbat

Option 2: Processing data using PSPP (Free)

PSPP is a free package; alternative to commercial software SPSS. We can use the same SPSS codes to read the datafile into PSPP, and save.

- **Step 1**: Get the free PSPP software from the website: www.gnu.org/software/pspp/

PSPP is available for GNU/Hurd, GNU/Linux, Darwin (Mac OS X), OpenBSD, NetBSD, FreeBSD, and Windows

For windows, download appropriate version.

pspp.awardspace.info

Highlights of the current PSPP-for-MSWindows setup

PP info:		**Package info:**	
rrent version:	Build from daily snapshot source, not fully tested	MSWindows version:	MSWindows7 and newer
estions/Suggestions:	pspp-users@gnu.org	Package Size:	40 Mb
ormation about PSPP:	https://www.gnu.org/software/pspp	Size on disk:	80 Mb
PP Manual:	PDF or HTML (will be installed on your PC by the installer package)	Technical:	MinGW based Cross-comp on openSUSE leap 42.3
w in latest build:	NEWS (open downloaded file with notepad)		

Downloads:

Installer for 32bits version Will work on 32 and 64bits MSWindows	Installer for 64bits version Works only on 64bits MSWindows	Source version close to a released version	Build generation
PSPP_2018-11-09_daily_32bits	PSPP_2018-11-09_daily_64bits	Yes 1.2.0-g0fb4db	42.2.3
PSPP_2017-09-09_daily_32bits	PSPP_2017-09-09_daily_64bits	Yes 1.0.1-g818227	42.2.3
PSPP_2017-07-30_daily_32bits	PSPP_2017-07-30_daily_64bits	Yes 0.10.5pre3-g9a68ff	42.2.3
PSPP_2016-09-27_daily_32bits	PSPP_2016-09-27_daily_64bits	No 0.10.4-g50f7b7	42.1.4

Download the file

Install

Click the icon shorcut after installing

- **Step 2**: Open PSPP

- **Step 3**: Go to 'file' menu and click 'open'

- **Step 4**: Specify the *readfile.sps* file from the 'SAS_SPSS' folder.

You will see the following file:

- **Step 5**: Similar to before, change the directories as appropriate. Get rid of the extra lines of codes. Resulting codes are as follows (you can copy and replace the code in the file with the following codes):

```
file handle infile/name = 'C:\CCHS\Data\cchs_cycle3-1CD\DATA\hs.txt'.
data list file = infile notable/.
include file = "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hs_i.sps".
include file = "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hsvale.sps".
include file = "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hsvare.sps".
include file = "C:\CCHS\Data\cchs_cycle3-1CD\SAS_SPSS\Layouts\hs\hsmiss.sps".
execute.
```

For Mac users, it should be as follows (e.g., `username` should be your user name, if you are saving under the path `"/Users/username/CCHS/Data/"`):

```
file handle infile/name ="/Users/username/CCHS/Data/cchs_cycle3-1CD/Data/hs.txt".
data list file = infile notable/.
include file = "/Users/username/CCHS/Data/cchs_cycle3-1CD/SAS_SPSS/Layouts/hs/hs_i.sps".
include file = "/Users/username/CCHS/Data/cchs_cycle3-1CD/SAS_SPSS/Layouts/hs/hsvale.sps".
include file = "/Users/username/CCHS/Data/cchs_cycle3-1CD/SAS_SPSS/Layouts/hs/hsvare.sps".
include file = "/Users/username/CCHS/Data/cchs_cycle3-1CD/SAS_SPSS/Layouts/hs/hsmiss.sps".

execute.
```

- **Step 6**: Run the codes.

- **Step 7**: This is a large data, and will take some time to load the data into the PSPP data editor. **Be patient**.

Once loading is complete, it will show the 'output' and 'data view'.

Note that, you will get error message, if your files were not in the correct path. In our example, the path was `"C:\CCHS\Data\"` for the zip file content (see the previous steps).

- **Step 7**: You can also check the 'variable view'.

- **Step 8**: Save the data by clicking 'File' and then 'save as ...'

- **Step 9**: Specify the name of the datafile and the location / folder to save the data file.

- **Step 10**: See the SAV file saved in the directory.

- **Step 11**: To save CSV format data, use the following syntax.

```
SAVE TRANSLATE
  /OUTFILE="C:/CCHS/Data/cchs3b.csv"
  /TYPE=CSV
  /FIELDNAMES
  /CELLS=VALUES.
```

Note that, for categorical data, you can either save values or labels. For our purpose, we prefer values, and hence saved with values here.

- **Step 12**: See the CSV file saved in the directory extracted from PSPP.

Option 3: Processing data using SPSS

Log into ubc.onthehub.com to download SPSS. With your CWL account, UBC students should be able to download it. UBC IT website for SPSS says:

```
The SPSS software license with UBC specifies that
SPSS must only be used by UBC Faculty, Students,
and Research Staff and only for Teaching and non-commercial
Research purposes related to UBC.
```

Both network (for UBC owened devices) or standalone / home versions (for non-UBC owened devices) should be available. Once downloaded, same process of importing CCHS data in PSPP can also be applied on SPSS (same syntax files should work).

Processing data in R

Download software

- **Step 1**: Download either 'R' from CRAN www.r-project.org or 'R open' from Microsoft mran.microsoft.com/open

- **Step 2**: Download RStudio from www.rstudio.com/

- **Step 3**: Open RStudio

Import, export and load data into R

- **Step 1**: Set working directory

```
setwd("C:/CCHS/Data/") # or something appropriate
```

- **Step 2**: Read the dataset created from PSPP with cell values. We can also do a small check to see if the cell values are visible. For example, we choose a variable 'CCCE_05A', and tabulate it.

```
Hs <- read.csv("cchs3b.csv", header = TRUE)
table(Hs$CCCE_05A)
```

- **Step 3**: Save the RData file from R into a folder `SurveyData`:

```
save(Hs, file = "SurveyData/cchs3.RData")
```

- **Step 4**: See the RData file saved in the directory extracted from R.

- **Step 5**: Close R / RStudio and restart it. Environment window within RStudio should be empty.

- **Step 6**: Load the saved RData into R. Environment window within RStudio should have 'Hs' dataset.

```
load("SurveyData/cchs3.RData")
```

Importing NHANES to R

This tutorial provides comprehensive instructions on accessing the National Health and Nutrition Examination Survey (NHANES) dataset from the US Centers for Disease Control and Prevention (CDC) website and importing it into the RStudio environment. It covers accessing NHANES Data:

- Directly from the CDC website: A step-by-step guide with accompanying images, illustrating how to navigate the CDC website, download the data, and interpret the accompanying codebook.
- Using R packages, specifically the nhanesA package: A concise guide on how to download and get summaries of the NHANES data using this R package.

```r
# Load required packages
#devtools::install_github("warnes/SASxport")
library(SASxport)
library(foreign)
library(nhanesA)
library(knitr)
require(DiagrammeR)
require(DiagrammeRsvg)
require(rsvg)
library(magrittr)
library(svglite)
library(png)
use.saved.chche <- TRUE
```

Before installing a package from GitHub, it's better to check whether you installed the right version of Rtools

Accessing NHANES Data Directly from the CDC website

In the following example, we will see how to download 'Demographics' data, and check associated variable in that dataset.

- **Step 1**: Say, for example, we are interested about the NHANES 2015-2016 survey. Clicking the associated link in the above Figure gets us to the page for the corresponding cycle (see below).

NHANES 1999-2000 and onward survey datasets are publicly available at wwwn.cdc.gov/nchs/nhanes/

NHANES 2015-2016

Contents in Detail

- Survey Questionnaires
- Examination and Laboratory Procedure Manuals
- Brochures and Consent Documents

Using the Data

- NHANES 2015-2016 Overview
- Technical Notes for Data Release
- Survey Methods and Analytic Guidelines
- Response Rates and Population Totals
- NHANES Web Tutorial

Data, Documentation, Codebooks, SAS Code

- Demographics
- Dietary
- Examination
- Laboratory
- Questionnaire
- Limited Access

- **Step 2**: There are various types of data available for this survey. Let's explore the demographic information from this cycle. These data are mostly available in the form of SAS `XPT` format (see below).

NHANES 2015-2016 Demographics Data

Data File Name	Doc File	Data File	Date Published
Demographic Variables and Sample Weights	DEMO_I Doc	DEMO_I Data [XPT - 3.6 MB]	September, 2017

- **Step 3**: We can download the XPT data in the local PC folder and read the data into R as as follows:

```
DEMO <- read.xport("Data/accessing/DEMO_I.XPT")
```

- **Step 4**: Once data is imported in RStudio, we will see the `DEMO` object listed under data window (see below):

- **Step 5**: We can also check the variable names in this `DEMO` dataset as follows:

```
names(DEMO)
#>  [1] "SEQN"     "SDDSRVYR" "RIDSTATR" "RIAGENDR" "RIDAGEYR" "RIDAGEMN"
#>  [7] "RIDRETH1" "RIDRETH3" "RIDEXMON" "RIDEXAGM" "DMQMILIZ" "DMQADFC"
#> [13] "DMDBORN4" "DMDCITZN" "DMDYRSUS" "DMDEDUC3" "DMDEDUC2" "DMDMARTL"
#> [19] "RIDEXPRG" "SIALANG"  "SIAPROXY" "SIAINTRP" "FIALANG"  "FIAPROXY"
#> [25] "FIAINTRP" "MIALANG"  "MIAPROXY" "MIAINTRP" "AIALANGA" "DMDHHSIZ"
#> [31] "DMDFMSIZ" "DMDHHSZA" "DMDHHSZB" "DMDHHSZE" "DMDHRGND" "DMDHRAGE"
#> [37] "DMDHRBR4" "DMDHREDU" "DMDHRMAR" "DMDHSEDU" "WTINT2YR" "WTMEC2YR"
#> [43] "SDMVPSU"  "SDMVSTRA" "INDHHIN2" "INDFMIN2" "INDFMPIR"
```

- **Step 6**: We can open the data in RStudio in the dataview window (by clicking the `DEMO` data from the data window). The next Figure shows only a few columns and rows from this large dataset. Note that there are some values marked as "NA", which represents missing values.

DMDHSEDU HH ref person's spouse's education level	WTINT2YR Full sample 2 year interview weight	WTMEC2YR Full sample 2 year MEC exam weight	SDMVPSU Masked variance pseudo-PSU	SDMVSTRA Masked variance pseudo-stratum
NA	9964.725	9860.625	1	120
5	44749.890	46173.307	2	124
NA	9891.944	10963.314	1	119
5	37043.087	39353.307	2	128
4	22744.355	23557.163	1	125
4	18526.160	18249.326	2	122
NA	20395.535	20068.663	2	126
NA	24788.723	25399.385	2	126
4	10998.012	11273.998	2	129
NA	34513.078	35673.964	1	121
NA	10988.317	11184.295	2	131
4	60125.441	63059.813	2	131
5	96194.928	97001.988	1	125
2	14862.011	14802.214	1	128

- **Step 7**: There is a column name associated with each column, e.g., `DMDHSEDU` in the first column in the above Figure. To understand what the column names mean in this Figure, we need to take a look at the codebook. To access codebook, click the `'DEMO|Doc'` link (in step 2). This will show the data documentation and associated codebook (see the next Figure).

TABLE OF CONTENTS

- Component Description
- Eligible Sample
- Interview Setting and Mode of Administration
- Quality Assurance & Quality Control
- Data Processing and Editing
- Analytic Notes
- References
- Codebook
  - SEQN - Respondent sequence number
  - SDDSRVYR - Data release cycle
  - RIDSTATR - Interview/Examination status
  - RIAGENDR - Gender
  - RIDAGEYR - Age in years at screening
  - DMDHRMAR - HH ref person's marital status
  - DMDHSEDU - HH ref person's spouse's education level
  - WTINT2YR - Full sample 2 year interview weight
  - WTMEC2YR - Full sample 2 year MEC exam weight

- **Step 8**: We can see a link for the column or variable `DMDHSEDU` in the table of content (in the above Figure). Clicking that link will provide us further information about what this variable means (see the next Figure).

DMDHSEDU - HH ref person's spouse's education level

Variable Name: DMDHSEDU

SAS Label: HH ref person's spouse's education level

English Text: HH reference person's spouse's education level

Target: Both males and females 0 YEARS - 150 YEARS

Code or Value	Value Description	Count	Cumulative	Skip to Item
1	Less Than 9th Grade	619	619	
2	9-11th Grade (Includes 12th grade with no diploma)	511	1130	
3	High School Grad/GED or Equivalent	980	2110	
4	Some College or AA degree	1462	3572	
5	College Graduate or above	1629	5201	
7	Refused	2	5203	
9	Don't Know	23	5226	
.	Missing	4745	9971	

- **Step 9**: We can assess if the numbers reported under count and cumulative (from the above Figure) matches with what we get from the `DEMO` data we just imported (particularly, for the `DMDHSEDU` variable):

```r
table(DEMO$DMDHSEDU) # Frequency table
#>
#>    1    2    3    4    5    7    9
#>  619  511  980 1462 1629    2   23
cumsum(table(DEMO$DMDHSEDU)) # Cumulative frequency table
#>    1    2    3    4    5    7    9
#>  619 1130 2110 3572 5201 5203 5226
length(is.na(DEMO$DMDHSEDU)) # Number of non-NA observations
#> [1] 9971
```

Accessing NHANES Data Using R Packages

nhanesA package

```r
library(nhanesA)
```

> **Tip**
>
> R package `nhanesA` provides a convenient way to download and analyze NHANES survey data.

- **Step 1**: Witin the CDC website, NHANES data are available in 5 categories
  - Demographics (`DEMO`)
  - Dietary (`DIET`)
  - Examination (`EXAM`)
  - Laboratory (`LAB`)
  - Questionnaire (`Q`)

RNHANES (Susmann 2016) is another packages for downloading the NHANES data easily.

To get a list of available variables within a data file, we run the following command (e.g., we check variable names within `DEMO` data):

```r
nhanesTables(data_group='DEMO', year=2015)
#>   Data.File.Name                    Data.File.Description
#> 1         DEMO_I Demographic Variables and Sample Weights
```

- **Step 2**: We can obtain the summaries of the downloaded data as follows (see below):

```r
demo <- nhanes('DEMO_I')
names(demo)
#>  [1] "SEQN"     "SDDSRVYR" "RIDSTATR" "RIAGENDR" "RIDAGEYR" "RIDAGEMN"
#>  [7] "RIDRETH1" "RIDRETH3" "RIDEXMON" "RIDEXAGM" "DMQMILIZ" "DMQADFC"
#> [13] "DMDBORN4" "DMDCITZN" "DMDYRSUS" "DMDEDUC3" "DMDEDUC2" "DMDMARTL"
#> [19] "RIDEXPRG" "SIALANG"  "SIAPROXY" "SIAINTRP" "FIALANG"  "FIAPROXY"
#> [25] "FIAINTRP" "MIALANG"  "MIAPROXY" "MIAINTRP" "AIALANGA" "DMDHHSIZ"
#> [31] "DMDFMSIZ" "DMDHHSZA" "DMDHHSZB" "DMDHHSZE" "DMDHRGND" "DMDHRAGE"
#> [37] "DMDHRBR4" "DMDHREDU" "DMDHRMAR" "DMDHSEDU" "WTINT2YR" "WTMEC2YR"
#> [43] "SDMVPSU"  "SDMVSTRA" "INDHHIN2" "INDFMIN2" "INDFMPIR"
table(demo$DMDHSEDU) # Frequency table
#>
#>    1    2    3    4    5    7    9
#>  619  511  980 1462 1629    2   23
cumsum(table(demo$DMDHSEDU)) # Cumulative frequency table
#>    1    2    3    4    5    7    9
#>  619 1130 2110 3572 5201 5203 5226
length(is.na(demo$DMDHSEDU)) # Number of non-NA observations
#> [1] 9971
```

Import data issue

Sometimes, you might see a warning message when downloading NHANES data using an R package. For example, simple-Warning in download.file(url, tf, mode = "wb", quiet = TRUE): cannot open URL, or 404 data Not Found.

The possible reason could be the NHANES server was down when you tried to connect. In that case, try later with the same codes. Also, check the name of the variables carefully. The name of the same variable could be different in different survey cycles. It is also possible that some variables are not available in all cycles.

References

Reproducing NHANES results

The section instructs on reproducing the results from a specific article, detailing the eligibility criteria and variables of interest, guiding the user through accessing, merging, and filtering relevant NHANES data, and then recoding and comparing the results to ensure they match with the original article's findings, all supported with visual aids and R code examples.

Example article

Let us use the article by Flegal et al. (2016) as our reference. DOI:10.1001/jama.2016.6458.

Flegal et al. (2016)

Task

Objectives are to

(1) Learn how to download and select pertinent NHANES data
(2) Understand the importance of cleaning and transforming the data
(3) Reproduce findings from an existing research paper using NHANES data

Our specific task in this tutorial is to **reproduce the numbers reported in Table 1** from this article.

Eligibility criteria

Methods section from this article says:

- "For adults aged 20 years or older, obesity was defined according to clinical guidelines."
- "Pregnant women were excluded from analysis."
- "Participant age was grouped into categories of 20 to 39 years, 40 to 59 years, and 60 years and older."
- Table 1 title says NHANES 2013-2014 was used.

Variables of interest

Before diving into NHANES, take some time to comprehend its structure. Detailed documentation provides crucial information about variables, age categories, and other specifics.

Variables of interest:

- `age` (eligibility and stratifying variable)
- `sex` (stratifying variable)
- `race` (stratifying variable)
- `pregnancy status` (eligibility)
- `obesity/BMI status` (main variable of interest for the paper)

Searching for necessary variables

Search these variables using the NHANES variable keyword search within the 2013-14 cycle: cdc.gov/nchs/nhanes/search/

- Below is an example for BMI variable search:

Search Term	Body Mass Index
Fields to Search	All
Sort By	Variable Name
Limited Access	Exclude
Release Cycle	2013-2014
Seach Result Page Size	50

Search

Number of variables found: 1

Variable Name	SAS Label	Variable Description	Data File Name	Component Link
BMXBMI	Body Mass Index (kg/m**2)	Body Mass Index (kg/m**2)	Body Measures (BMX_H)	2013-2014 Examination

- Identifying the component: Note that H is the index for 2013-14 cycle as seen in the picture:

NHANES 2013-2014 Examination Data

- NHANES 2013-2014 Examination Variable List
- Exam Procedure Manuals
- 2013-2014 Examination Data Overview
- SAS Universal Viewer
- Data User Agreement

Data File Name	Doc File	Data File	Date Published
Blood Pressure	BPX_H Doc	BPX_H Data [XPT - 1.7 MB]	October 2015
Body Measures	BMX_H Doc	BMX_H Data [XPT - 2 MB]	October 2015

- Identifying the variable:

wwwn.cdc.gov/Nchs/Nhanes/2013-2014/BMX_H.htm#BMXBMI

BMXBMI - Body Mass Index (kg/m**2)

Variable Name: BMXBMI
SAS Label: Body Mass Index (kg/m**2)
English Text: Body Mass Index (kg/m**2)
Target: Both males and females 2 YEARS - 150 YEARS

Code or Value	Value Description	Count	Cumulative	Skip to Item
12.1 to 82.9	Range of Values	9055	9055	
.	Missing	758	9813	

TABLE OF CONTENTS

- Codebook
  - SEQN - Respondent sequence number
  - BMDSTATS - Body Measures Component Status Code
  - BMXWT - Weight (kg)
  - BMIWT - Weight Comment
  - BMXRECUM - Recumbent Length (cm)
  - BMIRECUM - Recumbent Length Comment
  - BMXHEAD - Head Circumference (cm)
  - BMIHEAD - Head Circumference Comment
  - BMXHT - Standing Height (cm)
  - BMIHT - Standing Height Comment
  - BMXBMI - Body Mass Index (kg/m**2)

- Rest of the variables all coming from demographic component

Downloading relevant variables

You can download NHANES data directly from their website or use a package that allows easy access to NHANES data sets. For this tutorial, we'll be downloading data specifically from the 2013-2014 cycle, focusing on demographics and BMI metrics.

NHANES data often comes coded numerically for various categories, making it less straightforward to understand. Use the available translation functions to convert these codes into meaningful categories, easing the data interpretation process.

Demographic data

For the demographic data, we will use the `DEMO_H` file, where the index `H` represents the 2013-14 cycle.

Index `H` represents NHANES 2013-14 cycle

> **Tip**
>
> We use the `nhanes` function to download a NHANES datafile and `nhanesTranslate` function to encode the categorical variables to match with the CDC website.

```
library(nhanesA)
demo13 <- nhanes('DEMO_H')
Demo13 <- nhanesTranslate('DEMO_H', names(demo13), data=demo13)
#> No translation table is available for SEQN
#> Translated columns: RIDSTATR RIAGENDR RIDRETH1 RIDRETH3 RIDEXMON DMQMILIZ DMQADFC DMDBORN4
```

Examination data

We are using same H index for BMI.

```r
exam13 <- nhanes('BMX_H')
Exam13 <- nhanesTranslate('BMX_H', names(exam13), data=exam13)
#> No translation table is available for SEQN
#> Translated columns: BMDSTATS BMIWT BMIHT BMDBMIC BMDSADCM
```

See all the column names in the data

```r
names(Demo13)
#>  [1] "SEQN"     "SDDSRVYR" "RIDSTATR" "RIAGENDR" "RIDAGEYR" "RIDAGEMN"
#>  [7] "RIDRETH1" "RIDRETH3" "RIDEXMON" "RIDEXAGM" "DMQMILIZ" "DMQADFC"
#> [13] "DMDBORN4" "DMDCITZN" "DMDYRSUS" "DMDEDUC3" "DMDEDUC2" "DMDMARTL"
#> [19] "RIDEXPRG" "SIALANG"  "SIAPROXY" "SIAINTRP" "FIALANG"  "FIAPROXY"
#> [25] "FIAINTRP" "MIALANG"  "MIAPROXY" "MIAINTRP" "AIALANGA" "DMDHHSIZ"
#> [31] "DMDFMSIZ" "DMDHHSZA" "DMDHHSZB" "DMDHHSZE" "DMDHRGND" "DMDHRAGE"
#> [37] "DMDHRBR4" "DMDHREDU" "DMDHRMAR" "DMDHSEDU" "WTINT2YR" "WTMEC2YR"
#> [43] "SDMVPSU"  "SDMVSTRA" "INDHHIN2" "INDFMIN2" "INDFMPIR"
names(Exam13)
#>  [1] "SEQN"     "BMDSTATS" "BMXWT"    "BMIWT"    "BMXRECUM" "BMIRECUM"
#>  [7] "BMXHEAD"  "BMIHEAD"  "BMXHT"    "BMIHT"    "BMXBMI"   "BMDBMIC"
#> [13] "BMXLEG"   "BMILEG"   "BMXARML"  "BMIARML"  "BMXARMC"  "BMIARMC"
#> [19] "BMXWAIST" "BMIWAIST" "BMXSAD1"  "BMXSAD2"  "BMXSAD3"  "BMXSAD4"
#> [25] "BMDAVSAD" "BMDSADCM"
```

Retain only useful variables

```r
demo13select <- Demo13[c("SEQN", # Respondent sequence number
                         "RIDEXPRG", # Pregnancy status at exam
                         "RIAGENDR", # Gender
                         "RIDAGEYR", # Age in years at screening
                         "RIDRETH3")]  # Race/Ethnicity
exam13select <- Exam13[c("SEQN", # Respondent sequence number
                         "BMXBMI")] # Body Mass Index (kg/m**2)
```

Quick look at the data

```r
head(demo13select)
#>    SEQN RIDEXPRG RIAGENDR RIDAGEYR           RIDRETH3
#> 1 73557     <NA>     Male       69 Non-Hispanic Black
#> 2 73558     <NA>     Male       54 Non-Hispanic White
#> 3 73559     <NA>     Male       72 Non-Hispanic White
#> 4 73560     <NA>     Male        9 Non-Hispanic White
#> 5 73561     <NA>   Female       73 Non-Hispanic White
#> 6 73562     <NA>     Male       56   Mexican American
head(exam13select)
#>    SEQN BMXBMI
#> 1 73557   26.7
#> 2 73558   28.6
#> 3 73559   28.9
#> 4 73560   17.1
#> 5 73561   19.7
#> 6 73562   41.7
```

Merge data

You might find that the demographic data and BMI data are in separate files. In that case, you'll need to combine them using a common ID variable. Make sure the data aligns correctly during this process.

Use the ID variable `SEQN` to merge both data:

```r
merged.data <- merge(demo13select, exam13select, by = c("SEQN"),
                     all=TRUE)
head(merged.data)
#>    SEQN RIDEXPRG RIAGENDR RIDAGEYR           RIDRETH3 BMXBMI
#> 1 73557     <NA>     Male       69 Non-Hispanic Black   26.7
#> 2 73558     <NA>     Male       54 Non-Hispanic White   28.6
#> 3 73559     <NA>     Male       72 Non-Hispanic White   28.9
#> 4 73560     <NA>     Male        9 Non-Hispanic White   17.1
#> 5 73561     <NA>   Female       73 Non-Hispanic White   19.7
#> 6 73562     <NA>     Male       56   Mexican American   41.7
dim(merged.data)
```

Within NHANES datasets in a given cycle, each person has an unique identifier number (variable name `SEQN`). We can use this `SEQN` variable to merge their data.

```
#> [1] 10175     6
merged.data$SEQN <- NULL
```

Investigate merged data

Let's check whether any missing data available.

```
require(tableone)
#> Loading required package: tableone
tab_nhanes <- CreateTableOne(data=merged.data, includeNA = TRUE)
print(tab_nhanes, showAllLevels = TRUE)
#>
#>                        level                            Overall
#>   n                                                     10175
#>   RIDEXPRG (%)         Yes, positive lab pregnancy test    65 ( 0.6)
#>                        The participant was not pregnant  1150 (11.3)
#>                        Cannot ascertain if the particip    94 ( 0.9)
#>                        <NA>                              8866 (87.1)
#>   RIAGENDR (%)         Male                              5003 (49.2)
#>                        Female                            5172 (50.8)
#>   RIDAGEYR (mean (SD))                                  31.48 (24.42)
#>   RIDRETH3 (%)         Mexican American                  1730 (17.0)
#>                        Other Hispanic                     960 ( 9.4)
#>                        Non-Hispanic White                3674 (36.1)
#>                        Non-Hispanic Black                2267 (22.3)
#>                        Non-Hispanic Asian                1074 (10.6)
#>                        Other Race - Including Multi-Rac   470 ( 4.6)
#>   BMXBMI (mean (SD))                                    25.68 (7.96)
```

As we can see, the `RIDEXPRG` variable contains a huge amount of missing information.

```
summary(merged.data$BMXBMI)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>   12.10   19.70   24.70   25.68   30.20   82.90    1120
```

BMI also contains many missing values.

Applying eligibility criteria

We subset the data using criteria similar to the JAMA paper by Flegal et al. (2016) (see above)

Flegal et al. (2016)

```
# No missing BMI
analytic.data1 <- subset(merged.data, !is.na(BMXBMI))
dim(analytic.data1)
#> [1] 9055    5

# Age >= 20
analytic.data2 <- subset(analytic.data1, RIDAGEYR >= 20)
dim(analytic.data2)
#> [1] 5520    5


table(analytic.data2$RIDEXPRG,useNA = "always")
#>
#> Yes, positive lab pregnancy test The participant was not pregnant
#>                                65                             1143
#> Cannot ascertain if the particip                              <NA>
#>                                44                             4268
# Pregnant women excluded
analytic.data3 <-
  subset(analytic.data2, is.na(RIDEXPRG) |
           RIDEXPRG != "Yes, positive lab pregnancy test")
dim(analytic.data3)
#> [1] 5455    5
```

Recoding variables

Recode similar to the JAMA paper by Flegal et al. (2016) (see above)

Flegal et al. (2016)

```
analytic.data3$AgeCat <- cut(analytic.data3$RIDAGEYR,
                             c(0,20,40,60,Inf),
                             right = FALSE)
analytic.data3$Gender <- car::recode(analytic.data3$RIAGENDR,
```

```
                                  "'1'='Male'; '2'='Female'")
table(analytic.data3$Gender,useNA = "always")
#>
#> Female   Male   <NA>
#>   2817   2638      0
analytic.data3$Race <- car::recode(analytic.data3$RIDRETH3,
                                  "c('Mexican American',
                                  'Other Hispanic')='Hispanic';
                                  'Non-Hispanic White'='White';
                                  'Non-Hispanic Black'='Black';
                                  'Non-Hispanic Asian'='Asian';
                                  else=NA")
analytic.data3$Race <-
  factor(analytic.data3$Race,
         levels = c('White', 'Black', 'Asian', 'Hispanic'))
```

Reproducing Table 1

Cross-reference the variable names with the NHANES data dictionary to ensure they align with your research questions.

Let's now compare our table with with the Table 1 in the article:

```
# Dataset for males
analytic.data3m <- subset(analytic.data3, Gender == "Male")

## Dataset for females
analytic.data3f <- subset(analytic.data3,  Gender == "Female")

# Frequency table by age and gender
with(analytic.data3, table(AgeCat,Gender))
#>           Gender
#> AgeCat     Female Male
#>   [0,20)        0    0
#>   [20,40)     901  909
#>   [40,60)     999  897
#>   [60,Inf)    917  832
apply(with(analytic.data3, table(AgeCat,Gender)),1,sum)
#>   [0,20)  [20,40)  [40,60) [60,Inf)
```

```
#>        0     1810     1896     1749

# Frequency table by age and race
with(analytic.data3, table(AgeCat,Race))
#>           Race
#> AgeCat     White Black Asian Hispanic
#>   [0,20)       0     0     0        0
#>   [20,40)    734   362   216      412
#>   [40,60)    759   383   251      449
#>   [60,Inf)   850   370   156      353

# Frequency table by age and race for males
with(analytic.data3m, table(AgeCat,Race))
#>           Race
#> AgeCat     White Black Asian Hispanic
#>   [0,20)       0     0     0        0
#>   [20,40)    386   182   106      189
#>   [40,60)    360   179   120      215
#>   [60,Inf)   384   195    74      169

# Frequency table by age and race for females
with(analytic.data3f, table(AgeCat,Race))
#>           Race
#> AgeCat     White Black Asian Hispanic
#>   [0,20)       0     0     0        0
#>   [20,40)    348   180   110      223
#>   [40,60)    399   204   131      234
#>   [60,Inf)   466   175    82      184
```

As we can see, our frequencies exactly match with Table 1 in the article.

Also see (Dhana 2023) for a tidyverse solution

Tip

If your research aims to make population-level inferences, ensure that you also download sampling weights, stratum and cluster variables. These aren't mandatory for basic analysis but are crucial for population-level conclusions. We will explore more about this in the survey data analy-

sis chapter.

References

Importing NHIS to R

This tutorial provides instructions on accessing the National Health Interview Survey (NHIS) dataset from the US Centers for Disease Control and Prevention (CDC) website and importing it into the RStudio environment.

NHIS datafile and documents

The NHIS files are stored in the NHIS website in different formats. You can import this data in any statistical package that supports these file formats, e.g., ASCII, CSV, SAS.

In the recent NHIS (post-2019), data are available in 5 categories:

- Family file
- Household file
- Person file
- Sample adult file
- Sample child file

In the earlier NHIS (pre-2019), data are available in 8 categories:

- Interview data for adults
- Interview data for children
- Imputed income for adults
- Imputed income for children
- Imputed income
- Functioning and disability
- Paradata

Combining data

0.0.0.0.1 * Different cycles

It is possible to combine datasets from different years/cycles together in NHIS. Similar to NHANES, identification of the same person in NHIS across different cycles is not possible in the public release datasets. For appending data from different cycles, please make sure that the variable names/labels are the same/identical in years under consideration (in some years, names and labels do change).

0.0.0.0.2 * Within the same cycle

Within NHIS datasets in a given cycle, each sampled person has a household number (`HHX`), family number (`FMX`), and a person number within family (`FPX`). We can create a unique identifier based on these three variables and merge the datasets.

Accessing NHIS Data

Unlike NHANES where a R package is available to download the dataset, NHIS datasets need to be downloaded directly from the CDC website. In the following example, we will see how to download 'Adult' data from 2021 NHIS, and check associated variable in that dataset.

NHIS survey datasets are publicly available at https://www.cdc.gov/nchs/nhis/

- **Step 1**: Say, for example, we are interested to download the adult dataset in the CSV format:

- **Step 2**: We can download the data in the local PC folder, unzip it, and then read the data into R as as follows:

```r
adult21 <- read.csv("Data/accessing/adult21.csv", header = T)
```

- **Step 3**: Once data is imported in RStudio, we will see the `adult21` object listed under data window (see below):

- **Step 4**: We can check the variable names in this `adult21` dataset using the `names` function.

```r
names(adult21)
```

- **Step 5**: We can check how many unique adults are in this `adult21` dataset. Note that the `HHX` variable in the dataset is the unique household identifier, where only one adult per household was selected for interview. We can use this `HHX` variable to merge adult datafile with other datafiles (e.g., child data).

```r
length(unique(adult21$HHX))
#> [1] 29482
```

- **Step 6**: We can open the data in RStudio in the dataview window (by clicking the `adult21` data from the data window). The next Figure shows only a few columns and rows from this large dataset. Note that there are some values marked as "NA", which represents missing values.

Filter Cols: « < 1 - 50 > »

	URBRRL	RATCAT_A	IMPINCFLG_A	CVDVAC2YR_A	CVDVAC2MR_A	CVDVAC1YR_A	CVDVAC1MR_A	SHTCVD19AV_A
1	4	7	0	NA	NA	NA	NA	NA
2	4	12	0	NA	NA	NA	NA	NA
3	4	14	0	NA	NA	NA	NA	NA
4	3	11	0	NA	NA	NA	NA	NA
5	1	6	1	NA	NA	NA	NA	NA
6	1	6	1	NA	NA	NA	NA	NA
7	1	14	1	NA	NA	NA	NA	NA
8	1	14	0	NA	NA	NA	NA	NA
9	1	7	0	NA	NA	NA	NA	NA
10	1	14	0	NA	NA	NA	NA	NA
11	1	14	0	NA	NA	NA	NA	NA
12	1	13	0	NA	NA	NA	NA	NA
13	1	13	1	NA	NA	NA	NA	NA
14	1	14	1	NA	NA	NA	NA	NA
15	1	7	1	NA	NA	NA	NA	NA
16	3	14	0	NA	NA	NA	NA	NA
17	3	4	0	NA	NA	NA	NA	NA
18	3	6	1	NA	NA	NA	NA	NA

- **Step 7**: To understand what the column names mean in this Figure, we need to take a look at the codebook, which is also available on the CDC website:

Sample Adult Interview

- Summary [PDF – 305 KB]
- Codebook [PDF – 2.4 MB]
- Codebook for restricted-use variables [PDF – 664 KB]
- ASCII data file [ZIP – 3.4 MB]
- CSV data file [ZIP – 4.2 MB]
- SAS input statements [SAS – 168 KB]
- SPSS input statements [SPS – 196 KB]
- Stata input statements [DO – 123 KB]

- **Step 8**: We can see a check for the column or variables, e.g., `REGION`, in the codebook:

2021 NATIONAL HEALTH INTERVIEW SURVEY (NHIS)
Codebook for Sample Adult file (Document Version: 25 July 2022)
PUBLIC USE

Variable:	REGION
Module:	General
Section:	UCF : Unit Control File
File(s):	Adult, Child, Paradata
Data Type:	Numeric
Length:	1
Question Text:	
Description:	Household region
Recode:	
Universe:	HHX ne ' '
Universe Description:	All households
Sources:	
Question ID:	
Keywords:	
Notes:	
Evaluation Report:	

Unweighted frequencies:
REGION Household region

Code	Description	Frequency	Percent
1	Northeast	4775	16.20
2	Midwest	6327	21.46
3	South	10731	36.40
4	West	7649	25.94

Frequency Missing:

- **Step 9**: We can assess if the numbers reported under count matches with what we get from the `adult21` data we just imported (particularly, for the `REGION` variable):

```
# Frequency table
table(adult21$REGION, useNA = "always")
#>
#>     1     2     3     4  <NA>
#>  4775  6327 10731  7649     0
```

Similarly, we can download the child data and open it in R:

```
child21 <- read.csv("Data/accessing/child21.csv", header = T)
```

Let's check how many unique children are in this `child21` dataset:

```
length(unique(child21$HHX))
#> [1] 8261
```

Now let's check for the column or variables, e.g., `SEX_C`, in the codebook:

Variable:	SEX_C
Module:	Roster
Section:	HHC : Household Composition
File(s):	Child
Data Type:	Numeric
Length:	1
Question Text:	
Description:	Sex of Sample Child
Recode:	
Universe:	HHSTAT_C=1
Universe Description:	Sample children 0-17
Sources:	
Question ID:	
Keywords:	
Notes:	
Evaluation Report:	

Unweighted frequencies:

SEX_C Sex of Sample Child

Code	Description	Frequency	Percent
1	Male	4257	51.53
2	Female	4002	48.44
7	Refused	2	0.02
8	Not Ascertained	0	0.00
9	Don't Know	0	0.00

Frequency Missing:

We can assess if the numbers reported under count matches with what we get from the `child21` data we just imported:

```
# Frequency table
table(child21$SEX_C, useNA = "always")
#>
#>    1    2    7 <NA>
#> 4257 4002    2    0
```

Merging within the same cycle

> **Note**
>
> We can use `HHX` variable to merge different datafiles within the same survey cycle.

As mentioned earlier, `HHX` variable in the dataset is the unique household identifier. We can use this `HHX` variable to merge different datafiles within the same survey cycle. Say, we are interested in merging child age (AGEP_C) and sex (SEX_C) variables with the adult datafile. We can use the `merge` function as follows:

```
dat <- merge(adult21, child21[,c("HHX", "AGEP_C", "SEX_C")], by = "HHX", all = T)
```

As we can see, there are data from 30,673 unique households, suggesting that not all children are sampled from the same household of sampled adults.

```
length(unique(dat$HHX))
#> [1] 30673
```

Table 1

Now we will use the `adult21` dataset to create Table 1 with utilizing survey features (i.e., psu, strata, and weights). For that, let us create/recode some variables:

In a following chapter about survey data analysis, we will explain what these survey features mean.

```r
# Heart attack
adult21$heart.attack <- car::recode(adult21$MIEV_A, " 2 = 'No'; 1 = 'Yes'; else = NA",
                                    levels = c("No", "Yes"), as.factor = T)
table(adult21$heart.attack, useNA = "always")
#>
#>    No   Yes  <NA>
#> 28378  1078    26

# Diabetes
adult21$diabetes <- car::recode(adult21$DIBEV_A, " 2 = 'No'; 1 = 'Yes'; else = NA",
                                levels = c("No", "Yes"), as.factor = T)
table(adult21$diabetes, useNA = "always")
#>
#>    No   Yes  <NA>
#> 26318  3134    30

# Sex
adult21$sex <- car::recode(adult21$SEX_A, " '1'='Male'; '2'='Female'; else=NA",
                           levels = c("Female", "Male"), as.factor = T)
table(adult21$sex, useNA = "always")
#>
#> Female   Male   <NA>
#>  16102  13378      2

# Pseudo-PSU
adult21$psu <- adult21$PPSU
adult21$psu <- as.factor(adult21$psu)
table(adult21$psu, useNA = "always")
#>
#>    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
#> 1952 1599 1141  792  669  461  549  662  629  840  521  359  491  421  405  346
#>   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32
#>  156  221  348  531  596  656  588  593  383  647  374  294  218   57  310  334
#>   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48
#>  269  369  410  447  180  279  263   85   70  208  175  224  302  270  396  376
#>   49   50   51   52   53   54   55   56   57   58   59   60   61   62   63   64
#>  202  240  303  253  349  249   75   67  257  203  234  327  410  364  252  247
#>   65   66   67   68   72   73   74   75   76   77   78   79   80   81   82   87
#>  199  149   96   22   65   42  108  131   37   41   31   28   50   46   64   31
#>   89   90   91   92   93   97   98   99  100  101  102  103  104  108  109  110
```

```
#>   81   28  132  170   64   86   45   32  144  128  129  171  117   63   48   13
#>  114  127  128  134  139  140  150  151  152  153 <NA>
#>   54   44   10   29   69   18   46   50   49   24    0

# Pseudo-stratum
adult21$strata <- adult21$PSTRAT
adult21$strata <- as.factor(adult21$strata)
table(adult21$strata, useNA = "always")
#>
#>  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115
#>  736  714  589  480  499  605  733  748  757  629  623  614  158  914  386  603
#>  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131
#>  192  678  661  842  549  510  606  306  517  385  633  418  265  801  449  558
#>  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147
#>  558  563  434  532  595  576  494  370  644  485  460  738  625  368  412  650
#>  148  149  150  151 <NA>
#>  672  531  556 1061    0

# Sampling weight
adult21$sweight <- adult21$WTFA
summary(adult21$sweight)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   793.2  4698.3  7402.6  8586.9 10671.1 71378.0

# Drop the missing values associated with Heart attack, Diabetes, Sex
dat.analytic <- adult21[complete.cases(adult21$heart.attack),]
dat.analytic <- dat.analytic[complete.cases(dat.analytic$diabetes),]
dat.analytic <- dat.analytic[complete.cases(dat.analytic$sex),]
dim(dat.analytic)
#> [1] 29435   628
```

First, we will create the survey design. Second, we will report Table 1 with heart attack and sex variable, stratified by diabetes.

```
library(tableone)
library(survey)

# Indicator in the full data
adult21$indicator <- 1
```

```r
adult21$indicator[adult21$HHX %in% dat.analytic$HHX] <- 0
table(adult21$indicator)
#>
#>     0     1
#> 29435    47

# Survey design
w.design <- svydesign(id = ~psu, strata = ~strata, weights = ~sweight, data = adult21, nest = T

# Subset
w.design0 <- subset(w.design, indicator == 0)

# Table 1
tab1 <- svyCreateTableOne(var = c("heart.attack", "sex"), strata= "diabetes",
                          data = w.design0, test = FALSE)
print(tab1)
#>                          Stratified by diabetes
#>                           No                   Yes
#>   n                       228524605.2          24325386.4
#>   heart.attack = Yes (%)   5335722.7 ( 2.3)    2358175.6 ( 9.7)
#>   sex = Male (%)          109610086.2 (48.0)  12510287.1 (51.4)
```

Regression analysis

Let's run a regression analysis with utilizing survey features.

```r
library(Publish)

# Design-adjusted logistic
fit1 <- svyglm(I(heart.attack == "Yes") ~ diabetes + sex, design = w.design0, family = binomial
publish(fit1)
#>  Variable  Units OddsRatio       CI.95 p-value
#>  diabetes     No       Ref
#>               Yes      4.42 [3.76;5.21]  <1e-04
#>       sex Female       Ref
#>             Male      2.04 [1.75;2.39]  <1e-04
```

Linking mortality data

This tutorial provides instructions on linking public-use US mortality data with the NHANES dataset. One can also follow the same steps to link the mortality data with the NHIS.

Download mortality data

The public-use mortality data can be downloaded directly from the CDC website. Datasets are available in .dat format, separately for each cycle of NHANES and NHIS.

On the same website, CDC also provided R, SAS, and Stata codes with instructions on how to download the datasets directly from the website.

```
9146 R_ReadInProgramAllSurveys.R
8749 SAS_ReadInProgramAllSurveys.sas
9205 Stata_ReadInProgramAllSurveys.do
```

We can click on the desired survey link to download and save the datasets on our own hard drive. The dataset will be directly downloaded to our specified download folder. Alternatively, we can right-click on the desired survey link and select `Save link as...`

Note that the data file is saved as `<survey name>_MORT_2019_PUBLIC.dat`. In our example, we downloaded mortality data for the NHANES 2013-14 participants. Hence, the name of the file should be `NHANES_2013_2014_MORT_2019_PUBLIC.dat`.

Link mortality data to NHANES

Let us link the mortality data to the NHANES 2013-14 cycle. The steps are as follows:

- Download morality data for the NHANES 2013-14 cycle
- Load the morality data on the R environment
- Load NHANES 2013-14 cycle
- Merge two datasets using the unique identifier

Download morality data

We can follow the steps described above to download the mortality dataset directly from the CDC website.

Load the morality data on the R environment

To load the dataset, we can use the `read_fwf` function from the `readr` package.

```
library(readr)
library(dplyr)
#> Warning: package 'dplyr' was built under R version 4.2.3
#>
#> Attaching package: 'dplyr'
#> The following objects are masked from 'package:stats':
#>
#>     filter, lag
#> The following objects are masked from 'package:base':
#>
#>     intersect, setdiff, setequal, union

dat.mort <- read_fwf(
  file = "Data/accessing/NHANES_2013_2014_MORT_2019_PUBLIC.dat",
  col_types = "iiiiiiii",
  fwf_cols(SEQN = c(1,6),
           eligstat = c(15,15),
           mortstat = c(16,16),
           ucod_leading = c(17,19),
           diabetes = c(20,20),
           hyperten = c(21,21),
           permth_int = c(43,45),
           permth_exm = c(46,48)),
  na = c("", "."))

head(dat.mort)
#> # A tibble: 6 x 8
#>    SEQN eligstat mortstat ucod_leading diabetes hyperten permth_int permth_exm
#>   <int>    <int>    <int>        <int>    <int>    <int>      <int>      <int>
#> 1 73557        1        0           NA       NA       NA         78         78
```

```
#> 2 73558        1        1            7        1       0         58         58
#> 3 73559        1        0           NA       NA      NA         83         82
#> 4 73560        2       NA           NA       NA      NA         NA         NA
#> 5 73561        1        1           10        0       0         10          9
#> 6 73562        1        0           NA       NA      NA         68         68
```

In the code chuck above,

- SEQN: unique identifier for NHANES
- eligstat: Eligibility Status for Mortality Follow-up
  - 1 = Eligible
  - 2 = Under age 18, not available for public release
  - 3 = Ineligible
- mortstat: Mortality Status
  - 0 = Assumed alive
  - 1 = Assumed deceased
  - NA = Ineligible or under age 18
- ucod_leading: Underlying Cause of Death
  - 1 = Diseases of heart (I00-I09, I11, I13, I20-I51)
  - 2 = Malignant neoplasms (C00-C97)
  - 3 = Chronic lower respiratory diseases (J40-J47)
  - 4 = Accidents (unintentional injuries) (V01-X59, Y85-Y86)
  - 5 = Cerebrovascular diseases (I60-I69)
  - 6 = Alzheimer's disease (G30)
  - 7 = Diabetes mellitus (E10-E14)
  - 8 = Influenza and pneumonia (J09-J18)
  - 9 = Nephritis, nephrotic syndrome and nephrosis (N00-N07, N17-N19, N25-N27)
  - 10 = All other causes
  - NA = Ineligible, under age 18, assumed alive, or no cause of death data available
- diabetes: Diabetes Flag from Multiple Cause of Death (MCOD)

  - 0 = No - Condition not listed as a multiple cause of death
  - 1 = Yes - Condition listed as a multiple cause of death
  - NA = Assumed alive, under age 18, ineligible for mortality follow-up, or MCOD not available

- hyperten: Hypertension Flag from Multiple Cause of Death (MCOD)
  - 0 = No - Condition not listed as a multiple cause of death
  - 1 = Yes - Condition listed as a multiple cause of death
  - NA = Assumed alive, under age 18, ineligible for mortality follow-up, or MCOD not available

- permth_int: Person-Months of Follow-up from NHANES Interview date

- permth_exm: Person-Months of Follow-up from NHANES Mobile Examination Center (MEC) Date

Let us see the basic summary statistics of some variables:

```r
# Mortality Status
table(dat.mort$mortstat, useNA = "always")
#>
#>    0    1 <NA>
#> 5633  467 4075

# Person-Months of Follow-up from NHANES Interview date
summary(dat.mort$permth_int)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>    1.00   65.00   72.00   70.34   79.00   85.00    4075

# Underlying Cause of Death
table(dat.mort$ucod_leading, useNA = "always")
#>
#>    1    2    3    4    5    6    7    8    9   10 <NA>
#>  136   99   24   14   28   17   21    9   16  103 9708
```

Load NHANES 2013-14 cycle

Let the open the NHANES 2013-14 dataset we created in the previous chapter on Reproducing results.

```
# Load data
load("Data/accessing/analyticNHANES2013.RData")
ls()
#> [1] "analytic.data3"  "dat.mort"        "has_annotations" "merged.data"

# NHANES 2013-14
head(analytic.data3)
#>    SEQN RIDEXPRG RIAGENDR RIDAGEYR           RIDRETH3 WTINT2YR SDMVPSU SDMVSTRA
#> 1 73557     <NA>     Male       69 Non-Hispanic Black 13281.24       1      112
#> 2 73558     <NA>     Male       54 Non-Hispanic White 23682.06       1      108
#> 3 73559     <NA>     Male       72 Non-Hispanic White 57214.80       1      109
#> 5 73561     <NA>   Female       73 Non-Hispanic White 63709.67       2      116
#> 6 73562     <NA>     Male       56   Mexican American 24978.14       1      111
#> 8 73564     <NA>   Female       61 Non-Hispanic White 60325.10       1      114
#>   BMXBMI   AgeCat Gender     Race psu strata survey.weight
#> 1   26.7 [60,Inf)   Male    Black   1    112      13281.24
#> 2   28.6  [40,60)   Male    White   1    108      23682.06
#> 3   28.9 [60,Inf)   Male    White   1    109      57214.80
#> 5   19.7 [60,Inf) Female    White   2    116      63709.67
#> 6   41.7  [40,60)   Male Hispanic   1    111      24978.14
#> 8   35.7 [60,Inf) Female    White   1    114      60325.10
dim(analytic.data3)
#> [1] 5455   15
```

Merge mortality data and NHANES 2013-14 using unique identifier

Let us merge the mortality and NHANES datasets using the `SEQN` variable.

```
# Merge datasets
dat.nhanes <- merge(analytic.data3, dat.mort, by = "SEQN", all.x = T)
dim(dat.nhanes)
#> [1] 5455   22
head(dat.nhanes)
```

```
#>    SEQN RIDEXPRG RIAGENDR RIDAGEYR           RIDRETH3 WTINT2YR SDMVPSU SDMVSTRA
#> 1 73557     <NA>     Male       69 Non-Hispanic Black 13281.24       1      112
#> 2 73558     <NA>     Male       54 Non-Hispanic White 23682.06       1      108
#> 3 73559     <NA>     Male       72 Non-Hispanic White 57214.80       1      109
#> 4 73561     <NA>   Female       73 Non-Hispanic White 63709.67       2      116
#> 5 73562     <NA>     Male       56   Mexican American 24978.14       1      111
#> 6 73564     <NA>   Female       61 Non-Hispanic White 60325.10       1      114
#>   BMXBMI   AgeCat Gender     Race psu strata survey.weight eligstat mortstat
#> 1   26.7 [60,Inf)   Male    Black   1    112      13281.24        1        0
#> 2   28.6  [40,60)   Male    White   1    108      23682.06        1        1
#> 3   28.9 [60,Inf)   Male    White   1    109      57214.80        1        0
#> 4   19.7 [60,Inf) Female    White   2    116      63709.67        1        1
#> 5   41.7  [40,60)   Male Hispanic   1    111      24978.14        1        0
#> 6   35.7 [60,Inf) Female    White   1    114      60325.10        1        0
#>   ucod_leading diabetes hyperten permth_int permth_exm
#> 1           NA       NA       NA         78         78
#> 2            7        1        0         58         58
#> 3           NA       NA       NA         83         82
#> 4           10        0        0         10          9
#> 5           NA       NA       NA         68         68
#> 6           NA       NA       NA         63         62
```

Table 1

Now we will use the `dat.nhanes` dataset to create Table 1 with utilizing survey features (i.e., psu, strata, and survey weights). First, we will create the survey design. Second, we will report Table 1 with age, sex, race, eligibility, all-cause mortality status, diabetes-related death, hypertension-related death, and follow-up times.

```r
library(tableone)
library(survey)

# Make eligibility and mortality status as factor variable
factor.vars <- c("eligstat", "mortstat", "diabetes", "hyperten")
dat.nhanes[,factor.vars] <- lapply(dat.nhanes[,factor.vars] , factor)

# Survey design
```

```r
w.design <- svydesign(id = ~psu, strata = ~strata, weights = ~survey.weight,
                      data = dat.nhanes, nest = T)

# Table 1 - unweighted frequency or mean
tab1a <- CreateTableOne(var = c("AgeCat", "Gender", "Race", "eligstat", "mortstat",
                                "diabetes", "hyperten", "permth_int", "permth_exm"),
                        data = dat.nhanes, includeNA = T)
print(tab1a, showAllLevels = T, format = "f")
#>
#>                          level    Overall
#>   n                               5455
#>   AgeCat                 [0,20)       0
#>                          [20,40)   1810
#>                          [40,60)   1896
#>                          [60,Inf)  1749
#>   Gender                 Female    2817
#>                          Male      2638
#>   Race                   White     2343
#>                          Black     1115
#>                          Asian      623
#>                          Hispanic  1214
#>                          <NA>       160
#>   eligstat               1         5445
#>                          3           10
#>   mortstat               0         5030
#>                          1          415
#>                          <NA>        10
#>   diabetes               0          374
#>                          1           41
#>                          <NA>      5040
#>   hyperten               0          344
#>                          1           71
#>                          <NA>      5040
#>   permth_int (mean (SD))          70.40 (12.18)
#>   permth_exm (mean (SD))          69.49 (12.20)

# Table 1 - weighted percentage or mean
tab1b <- svyCreateTableOne(var = c("AgeCat", "Gender", "Race", "eligstat", "mortstat",
                                "diabetes", "hyperten", "permth_int", "permth_exm"),
                           data = w.design, includeNA = T)
```

```r
print(tab1b, showAllLevels = T, format = "p")
#>
#>                          level    Overall
#>   n                                217464332.1
#>   AgeCat (%)             [0,20)            0.0
#>                          [20,40)          35.5
#>                          [40,60)          37.5
#>                          [60,Inf)         27.0
#>   Gender (%)             Female           51.4
#>                          Male             48.6
#>   Race (%)               White            66.1
#>                          Black            11.4
#>                          Asian             5.2
#>                          Hispanic         14.7
#>                          <NA>              2.7
#>   eligstat (%)           1                99.9
#>                          3                 0.1
#>   mortstat (%)           0                93.6
#>                          1                 6.2
#>                          <NA>              0.1
#>   diabetes (%)           0                 5.5
#>                          1                 0.7
#>                          <NA>             93.8
#>   hyperten (%)           0                 5.2
#>                          1                 1.0
#>                          <NA>             93.8
#>   permth_int (mean (SD))                 70.71 (11.28)
#>   permth_exm (mean (SD))                 69.80 (11.31)
```

R Functions (A)

The section introduces a set of R functions useful for accessing and processing complex survey data, providing their descriptions and the packages they belong to.

Function_name	Package_name	Description
apply	base	Applies a function over an array or matrix.
cut	base	Converts a numeric variable to a factor variable.
merge	base/data.table	Merges multiple datasets.
names	base	Retrieves the names of an object.
nhanes	nhanesA	Downloads a NHANES datafile.
nhanesTables	nhanesA	Lists available variables within a datafile.
nhanesTranslate	nhanesA	Encodes categorical variables to match with certain standards, e.g., CDC we
recode	car	Recodes a variable.

For more information, visit the resources mentioned earlier.

Quiz (A)

> ⚠ Warning
>
> We are currently in the beta testing phase for our quiz page updates, so please be aware that some responses may not be accurate. Thank you for your patience.
> For those interested in downloading the quiz questions, you'll find a section below dedicated to downloading.

Live Quiz

You can click on the live quiz options to either see hints of why your response may be incorrect (response marked in red), or whether your choice of response is correct (response marked in green):

Download Quiz

1. **Downloading the File:**
    - Navigate to the link: See here. If not downloading immediately, right-click on the link and select "Save link as…" from the dropdown menu.
    - Alternatively click here for downloading the quizzes.
    - Choose a destination folder on your computer where you'd like to save the file (e.g., Desktop). **Remember this location**, as you'll need to navigate to it later.
2. **Setting Up RStudio:**

- If you don't have RStudio installed, see the download link in here.
- Launch RStudio after installing.

3. **Installing Necessary R Packages:**

   - Before running the R Markdown file, ensure you have the required packages. In RStudio's console (located at the bottom left by default), enter the following commands to install them:

```
install.packages("learnr")
install.packages("xfun")
```

4. **Opening and Running the File:**

   - In RStudio, go to `File` > `Open File...`.
   - Navigate to the folder where you saved the `Rmd` file and select it to open.
   - Once the file is open in RStudio, you'll see a "Run Document" button (green) at the top of the script editor. Click on it to run the R Markdown Quiz file.

App (A)

Below is an example of an app that utilizes NHANES demographic datasets following the tutorial materials. Users can tabulate and visualize the data summaries from the downloaded data from a selected NHANES cycle.

> Tip
>
> **Web App may need some time to load the necessary files, depending on your internet speed.**

> Important
>
> If you want to run the app locally in your computer instead for faster load time: you can download the app directly in your R via `shiny::runGitHub("EpiShinyLiveA", "ehsanx", subdir = "app")`. Make sure that `shiny`, `nhanesA`, `DataExplorer`, `Hmisc`, `dplyr`, and `tableone` packages are installed in your computer/R.

> Warning
>
> **Bug Report**:
> Fill out this form to report any issues with the app.

Exercise (A)

> Tip
>
> You cannot input any codes on this website but need to download the exercise files on your computer. You can download all of the related files in a zip file **accessingEx.zip** from Github folder, or just by clicking this link directly.
>
> - Navigate to the GitHub folder (above link) where the ZIP file is located.
> - Click on the file name (above zip file) to open its preview window.
> - Click on the Download button to download the file. If you can't see the Download button, click on "Download Raw File" link that should appear on the page.

Problem Statement

We will use the article by Palis, Marchand, and Oviedo-Joekes (2020), DOI: 10.1080/09638237.2018.1437602.

- Download the CCHS MH topical index
- Download the CCHS MH Data Dictionary

Question 1: [60% grade]

1(a) Importing dataset

```r
# Importing dataset
load("Data/accessing/cchsMH.RData")
```

1(b) Subsetting according to eligibility

Subset the dataset according to the eligibility criteria / restriction specified in the paper

- Identify the variable needed for eligibility criteria

Hint

- Read the first paragraph of **Analytic sample** (page 2) for the eligibility criteria
- Eligibility criteria was determined based on only **one** variable. Only work with 'YES' category.

```r
# your code here
```

1(c) Retaining necessary variables

In the dataset, retain only the variables associated with outcome measure, explanatory variable, potential confounders and survey weight. There should be eight variables (one outcome, one exposure, five confounders, and one survey weight).

Here are the steps:

- Identify the outcome variable
- Identify the explanatory variable
- Identify the potential confounders
- Identify the survey weight variable
- **Hint**

1. Read

- first and second paragraphs of **Study variables** for the outcome, explanatory and confounding variables
- third paragraph of the **Statistical analyses** for the survey weights variable.

2. There were **five** potential confounders.
3. Potentially useful functions for this exercise:

- %in%
- levels
- recode
- subset
- as.factor
- relevel
- or dplyr ways: filter, select

```
# your code here
```

1(d) Creating analytic dataset

Outcome variable has a category 'NOT STATED', but for our analysis, we will omit anyone associated with this category. Similarly, for explanatory variable, we have categories such as DON'T KNOW, REFUSAL and NOT STATED. We will omit anyone with these categories.

- Assign missing values for categories such as DON'T KNOW, REFUSAL and NOT STATED.
- Recode the variables as shown in Table 1 in the article. You can use any function/package of your choice. Here is an example (but feel free to use other functions. In R there are many other ways to do this same task.

```
## your code here
# levels(your.data.frame$your.age.variable) <-
#   list("15 to 24 years" = c("15 TO 19 YEARS", "20 TO 24 YEARS"),
#        "25 to 34 years" = c("25 TO 29 YEARS", "30 TO 34 YEARS"),
#        "35 to 44 years" = c("35 TO 39 YEARS", "40 TO 44 YEARS"),
#        "45 to 54 years" = c("45 TO 49 YEARS", "50 TO 54 YEARS"),
```

```
#         "55 to 64 years" = c("55 TO 59 YEARS", "60 TO 64 YEARS"),
#         "65 years or older" = c("65 TO 69 YEARS", "70 TO 74 YEARS",
#         "75 TO 79 YEARS", "80 YEARS OR MORE"))
```

1(e) Number of columns and variable names

Report the number of columns in your analytic dataset, and the variable names.

```
# your code here
```

Question 2: Table 1 [20% grade]

Reproduce Table 1 presented in the article (or see below). Omit the 'Main source of income' variable from the table. The table you produce should report numbers as follows, with all columns as shown in the table. In other words, the numbers should match.

Self-rated Mental Health Variable	Total n(%)	Poor or Fair n(%)	Good n(%)	Very good or excellent n(%)
Study sample	2628 (100)	1002 (38.1)	885 (33.7)	741 (28.2)
Community belonging				
- Very weak	480 (18.3)	282 (28.1)	118 (13.3)a	80 (10.8)a
- Somewhat weak	857 (32.6)	358 (35.7)	309 (34.9)	190 (25.6)
- Somewhat strong	1005 (38.2)	288 (28.7)	362 (40.9)	355 (47.9)
- Very strong	286 (10.9)	74 (7.4)a	96 (10.8)a	116 (15.7)a
Sex				
- Females	1407 (53.5)	616 (61.5)	487 (55.0)	304 (41.0)

Self-rated Mental Health Variable	Total n(%)	Poor or Fair n(%)	Good n(%)	Very good or excellent n(%)
- Males	1221 (46.5)	386 (38.5)	398 (45.0)	437 (59.0)
Age group				
- 15 to 24 years	740 (28.2)	191 (19.1)	264 (29.8)	285 (38.5)
- 25 to 34 years	475 (18.1)	141 (14.1)	167 (18.9)	167 (22.5)
- 35 to 44 years	393 (15.0)	185 (18.5)	119 (13.4)a	89 (12.0)a
- 45 to 54 years	438 (16.6)	220 (22.0)	139 (15.7)	79 (10.7)a
- 55 to 64 years	379 (14.4)	198 (19.7)	113 (12.8)a	68 (9.2)a
- 65 years or older	203 (7.7)	67 (6.6)a	83 (8.4)a	53 (7.1)b
Race/Ethnicity				
- Non-white	458 (17.4)	184 (18.4)	140 (15.8)	134 (18.1)
- White	2170 (82.6)	818 (81.6)	745 (84.2)	607 (81.9)
Main source of income				
- Employment Income^d	1054 (40.1)	289 (28.8)	386 (43.6)	379 (51.1)
- Worker's Compensation^e	160 (6.1)	91 (9.1)a	44 (5.0)b	25 (3.4)c
- Senior Benefits^f	134 (5.1)	57 (5.7)a	42 (4.7)b	35 (4.7)
- Other^g	184 (7.0)	82 (8.2)a	60 (6.8)a	42 (5.7)b
- Not applicable^h	851 (32.4)	402 (40.1)	263 (29.7)	186 (25.1)
- Not Stated^i	245 (9.3)	81 (8.1)a	90 (10.2)a	74 (10.0)

[a] Coefficient of variation between 16.6 and 25.0%. [b] Coefficient of variation between 25.1 and 33.3%. [c] Coefficient of variation > 33.3%. [d] Employment Income: Wages/salaries or self-employment. [e] Worker's compensation: Employment insurance or worker's compensation or social assistance/welfare. [f] Senior Benefits: Benefits from Canada or Quebec Pension Plan or job related retirement pensions, superannuation and annuities or RRSP/RRIF of Old Age Security and Guaranteed Income Supplement. [g] Other: Dividends/interest or child tax benefit or child support or alimony or other or no income. [h] Not applicable: Respondents who live in a household with only one person. The income variable "main source of personal income" is applicable only to those that live in a household of more than one person. [i] Not Stated: Question was not answered (don't know, refusal, not stated).

```r
# your code here
require(tableone)
```

Question 3: [20% grade]

3(a) Subset

Subset the dataset excluding 'Very good or excellent' responses from the self-rated mental health variable

```r
# your code here
```

3(b) Recode

Recode self-rated mental health variable and make it a binary variable: 'Good' vs. 'Poor' (simplifying category labels only). Convert that variable to a factor variable with 'Poor' being the reference level.

```r
# your code here
```

3(c) Regression

Run a logistic regression model for finding the relationship between community belonging (Reference: Very weak) and self-rated mental health (Reference: Poor) among respondents with mental or substance use disorders. Adjust the model for three confounders: sex, age, and race/ethnicity. Do not need to report summary of the model.

```
# your code here
```

3(d) Reporting odds ratio

Report the odds ratios and associated confidence intervals. Publish or jtools package could be useful to report the odds ratios with confidence intervals.

```
# your code here
```

Part III

Research questions

Background

When we are starting a research project, one of the first steps is to clearly define your research topic or question. We will primarily focus on two types of research questions:

(a) predictive (predictors predicting one outcome)
(b) associational or causal (association between an outcome and an exposure, adjusting for confounders and risk factors for the outcome).

> ❗ Important
>
> **Datasets**:
> All of the datasets used in this tutorial can be accessed from this GitHub repository folder

Overview of tutorials

Predictive questions

RHC Data

The first tutorial serves to educate the user on how to utilize the RHC dataset to answer a **predictive** research question: developing a prediction model for the length of stay. The tutorial equips users with the skills to clean and process raw data, transforming it into an analyzable format, and introduces concepts that will be foundational for subsequent analysis.

Data from NHANES Part 1: prepare data Part 2: work with data

The second tutorial (part a for downloading and part b for analyzing) provides an in-depth guide on how to build a **predictive** model for Diastolic blood pressure using the NHANES dataset for the years 2013-14.

In the previous chapter, we learned about how to access external data. In this chapter, we will embark on a journey to understand the nuances of different research questions, laying the groundwork for the topics that lie ahead. As we move forward, the next chapter will delve deeper into the challenges associated with causal questions. We will explore the complexities of causal associations and discuss the optimal types of variables to include in adjustment models for accurate treatment effect estimation. Following that, we will transition to a chapter dedicated entirely to predictive questions, shedding light on their unique attributes and the methodologies best suited for addressing them. Join us as we navigate these intricate terrains of research inquiry.

Causal questions

Data from CCHS

The third tutorial aims to guide a study on the relationship between Osteoarthritis (OA) and cardiovascular diseases (CVD) among Canadian adults from 2001-2005. Utilizing the Canadian Community Health Survey (CCHS) cycle 1.1-3.1, the study intends to explore whether OA **increases** (more accurately, whether **associated** with) the risk of developing CVD.

Data from NHANES

The NHANES dataset was analyzed in this fourth tutorial to explore the relationship between health predictors and cholesterol levels (**association/causal**). After refining the survey design and handling missing data, regression models were built using varying predictors. Standard error computations and p-values were derived, adjusting for the survey's unique structure.

> Tip
>
> **Optional Content**:
> You will find that some sections conclude with an optional video walkthrough that demonstrates the code. Keep in mind that the content might have been updated since these videos were recorded. Watching these videos is optional.

> Warning
>
> **Bug Report**:
> Fill out this form to report any issues with the tutorial.

Reference

Concepts (Q)

According to Thabane et al., there is a structured approach to do this, primarily using the PICOT and FINER frameworks.

Thabane et al. (2009)

PICOT Framework

The PICOT framework helps to structure a specific and clear research question by focusing on five key elements:

Research Question: "In US adults, does having rheumatoid arthritis, compared to those without rheumatoid arthritis, affect the rate of cardiovascular diseases during 1999–2018?" based on Hossain et al. (2022): DOI: 10.1016/j.annepidem.2022.03.005

FINER Criteria

Once we have formulated your research question with the help of the PICOT elements, we should evaluate it using the FINER criteria:

The key takeaway is: Use the PICOT and FINER frameworks to guide you in framing a compelling, ethical, and achievable research question.

Element	Description	Example
P	Population of Interest: Who is the target group you are studying?	US adults
I	Intervention: What is the main action, treatment, or variable you're looking at?	Effect of h
C	Comparison: Are you comparing the intervention against a control group or usual care?	People wit
O	Outcome of Interest: What specifically do you want to measure?	Rate of ca
T	Time Frame: Over what time period will your study take place?	1999–2018

Table 10: FINER Criteria

Element	Description
F	Feasible: Is it possible to conduct this research with available resources?
I	Interesting: Is the research question intriguing to the scientific community?
N	Novel: Is the question original and not already thoroughly researched?
E	Ethical: Is the research ethically sound?
R	Relevant: Is the research currently needed or will it fill a gap in existing knowledge?

SAP

A Statistical Analysis Plan (SAP), also referred to as a Data Analysis Plan (DAP) or Reporting Analysis Plan (RAP), is an integral part of research, particularly in randomized controlled trials (RCTs) (Kahan et al. 2020), but also in observational studies (Hiemstra et al. 2019). Here are a few reasons why it is beneficial to pre-plan the SAP for an observational study:

1. Pre-planning an SAP helps define the specific analytical strategies and methods that will be used to answer the research questions. It outlines the techniques for handling data, including

- the treatment of missing data, outliers,
- the use of statistical tests, and
- confounding adjustment techniques.

2. By detailing the analysis plan before the data is examined, researchers ensure transparency and reduce the risk of data dredging or p-hacking.
3. Confounding is a more pronounced issue in observational studies. Strategies for addressing confounding need to be more elaborate and explicit in observational studies.

> **Note**
>
> We include 2 types of tutorials that emphasize the critical steps of **data preparation** and analysis tailored to specific research questions, cosidering the PICOT frame-

Refer to the 'Scientific Writing for Health Research' book chapter for more details and examples for PICOT, FINER and Statistical Analysis Plan (SAP).

work. They underscore the importance of refining and cleaning datasets to ensure their suitability for rigorous analytical procedures. The analyses, while rooted in distinct methodologies, converge on the common goal of deriving meaningful insights and ensuring the integrity and validity of the results obtained from the processed analytical data.

Data preparation: Merging, reformatting and recategorizing essential variables to create a dataset suitable for analysis, aligning it with the study's objectives.

Video Lessons

PICOT and FINER

What is included in this Video Lesson:

- References 0:53
- How to get an idea about a Research Question? 1:05
- Why the question need to be good? 2:41
- A framework for defining a research question 5:17
- Think hard about the 'Outcome' 14:40
- Is this research doable? 17:57
- Overall Roadmap 19:57
- Other Reference (optional) 21:27

The timestamps are also included in the YouTube video description.

SAP

What is included in this Video Lesson:

- SAP 0:03
- SAP example from a RCT 1:31
- SAP example from an observational study 4:40
- Code book 15:35

The timestamps are also included in the YouTube video description.

Video Lesson Slides

Links

- Google Slides
- PDF Slides

References

Predictive question-1

```r
# Load required packages
require(tableone)
require(Publish)
require(MatchIt)
require(cobalt)
require(ggplot2)
```

Working with a Predictive question using RHC

This tutorial delves into processing and understanding the right heart catheterization (RHC) dataset, which pertains to patients in the intensive care unit. The dataset is particularly centered around the implications of using RHC in the early phases of care, with a focus on comparing two patient groups: those who received the RHC procedure and those who did not. The key outcome being analyzed is the 30-day survival rate. We will use this as an example to explain how to work with a predictive research question to build the analytic data.

(Connors et al. 1996) published an article in JAMA. The article is about managing or guiding therapy for the critically ill patients in the intensive care unit. They considered a number of **health-outcomes** such as

Link for the RHC dataset

- *length of stay* (hospital stay; measured continuously)
- *death* within certain period (death at any time up to 180 Days; measured as a binary variable)

The original article was concerned about the association of right heart catheterization *(RHC)* use during the first 24 hours of care in the intensive care unit and the health-outcomes mentioned above.

But we will use this data as a case study for our **prediction modelling**. Traditional PICOT framework is designed primarily for clinical questions related to interventions, so when applying it to other areas like predictive modeling, some creative adaptation is needed.

Aspect	Description
P	Patients who are critically ill
I	Not applicable, as we are dealing with a prediction model here
C	Not applicable, as we are dealing with a prediction model here
O	in-hospital mortality
T	Between 1989 to 1994 (see the JAMA paper)

We are interested in developing a prediction model for the length of stay.

Data download

Data is freely available from Vanderbilt Biostatistics, variable list is available here, and the article is freely available from researchgate.

Let us download the dataset and save it for later use.

Variable list and search for right heart catheterization dataset

```
# Load the dataset
ObsData <- read.csv("https://hbiostat.org/data/repo/rhc.csv",
                    header = TRUE)

# Save the dataset
saveRDS(ObsData, file = "Data/researchquestion/rhc.RDS")
```

Creating analytic dataset

Now, we show the process of preparing our analytic dataset (i.e., ready to use dataset for our analysis), so that the variables generally match with the way the authors were coded in the original article. Below we show the process of creating the analytic dataset.

Add column for outcome: length of stay

```r
# Length of Stay = date of discharge - study admission date
ObsData$Length.of.Stay <- ObsData$dschdte - ObsData$sadmdte

# Length of Stay = date of death - study admission date if date of discharge not available
ObsData$Length.of.Stay[is.na(ObsData$Length.of.Stay)] <-
  ObsData$dthdte[is.na(ObsData$Length.of.Stay)] -
  ObsData$sadmdte[is.na(ObsData$Length.of.Stay)]
```

Recoding column for outcome: death

> **Tip**
>
> Here we use the `ifelse` function to create a categorical variable. Other related functions are `cut`, `car`.

Let us recode our outcome variable as a binary variable:

```r
ObsData$death <- ifelse(ObsData$death == "Yes", 1, 0)
```

Remove unnecessary outcomes

Our next task is to remove unnecessary outcomes:

> **Tip**
>
> There are multiple ways to drop variables from a dataset. E.g., without using any package and using the `select` function from the dplyr package.

```r
ObsData <- dplyr::select(ObsData, !c(dthdte, lstctdte, dschdte,
                                     t3d30, dth30, surv2md1))
```

Remove unnecessary and problematic variables

Now we will drop unnecessary and problematic variables:

```
ObsData <- dplyr::select(ObsData, !c(sadmdte, ptid, X, adld3p,
                                     urin1, cat2))
```

Basic data cleanup

Now we will do some basic cleanup.

> **Tip**
>
> We an use the `lapply` function to convert all categorical variables to factors at once. Not that a similar function to `lapply` is `sapply`. The main difference is that sapply attempts to convert the result into a vector or matrix, while lapply returns a list.

```
# convert all categorical variables to factors
factors <- c("cat1", "ca", "death", "cardiohx", "chfhx",
             "dementhx", "psychhx", "chrpulhx", "renalhx",
             "liverhx", "gibledhx", "malighx", "immunhx",
             "transhx", "amihx", "sex", "dnr1", "ninsclas",
             "resp", "card", "neuro", "gastr", "renal", "meta",
             "hema", "seps", "trauma", "ortho", "race",
             "income")
ObsData[factors] <- lapply(ObsData[factors], as.factor)

# convert RHC.use (RHC vs. No RHC) to a binary variable
ObsData$RHC.use <- ifelse(ObsData$swang1 == "RHC", 1, 0)
ObsData <- dplyr::select(ObsData, !swang1)

# Categorize the variables to match with the original paper
ObsData$age <- cut(ObsData$age, breaks=c(-Inf, 50, 60, 70, 80, Inf),
                   right=FALSE)
ObsData$race <- factor(ObsData$race,
                       levels=c("white","black","other"))
ObsData$sex <- as.factor(ObsData$sex)
```

```r
ObsData$sex <- relevel(ObsData$sex, ref = "Male")
ObsData$cat1 <- as.factor(ObsData$cat1)
levels(ObsData$cat1) <- c("ARF","CHF","Other","Other","Other",
                          "Other","Other","MOSF","MOSF")
ObsData$ca <- as.factor(ObsData$ca)
levels(ObsData$ca) <- c("Metastatic","None","Localized (Yes)")
ObsData$ca <- factor(ObsData$ca, levels=c("None", "Localized (Yes)",
                                          "Metastatic"))
```

Rename variables

```r
# Rename the variables
names(ObsData) <- c("Disease.category", "Cancer", "Death",
                    "Cardiovascular", "Congestive.HF",
                    "Dementia", "Psychiatric", "Pulmonary",
                    "Renal", "Hepatic", "GI.Bleed", "Tumor",
                    "Immunosupperssion", "Transfer.hx", "MI",
                    "age", "sex", "edu", "DASIndex",
                    "APACHE.score", "Glasgow.Coma.Score",
                    "blood.pressure", "WBC", "Heart.rate",
                    "Respiratory.rate",  "Temperature",
                    "PaO2vs.FIO2", "Albumin", "Hematocrit",
                    "Bilirubin", "Creatinine", "Sodium",
                    "Potassium", "PaCo2",  "PH", "Weight",
                    "DNR.status", "Medical.insurance",
                    "Respiratory.Diag", "Cardiovascular.Diag",
                    "Neurological.Diag", "Gastrointestinal.Diag",
                    "Renal.Diag", "Metabolic.Diag",
                    "Hematologic.Diag", "Sepsis.Diag",
                    "Trauma.Diag", "Orthopedic.Diag",
                    "race", "income",
                    "Length.of.Stay", "RHC.use")

# Save the dataset
saveRDS(ObsData, file = "Data/researchquestion/rhcAnalytic.RDS")
```

Notations

let us introduce with some notations:

Notations	Example in RHC study
Y_1: Observed outcome	length of stay
Y_2: Observed outcome	death within 3 months
L: Covariates	See below

Basic data exploration

Dimension

Let us the how many rows and columns we have:

```
dim(ObsData)
#> [1] 5735   52
```

Comprehensive summary

Let us see the summary statistics of the variables:

> **Tip**
>
> To see the comprehensive summary of the variables, we can use the `skim` function form skimr package or `describe` function from rms package

```
require(skimr)
#> Loading required package: skimr
#> Warning: package 'skimr' was built under R version 4.2.3
skim(ObsData)
```

Table 13: Data summary

Name	ObsData
Number of rows	5735

Number of columns	52
Column type frequency:	
factor	31
numeric	21
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
Disease.category	0	1	FALSE	4	ARF: 2490, MOS: 1626, Oth: 1163, CHF: 456
Cancer	0	1	FALSE	3	Non: 4379, Loc: 972, Met: 384
Death	0	1	FALSE	2	1: 3722, 0: 2013
Cardiovascular	0	1	FALSE	2	0: 4722, 1: 1013
Congestive.HF	0	1	FALSE	2	0: 4714, 1: 1021
Dementia	0	1	FALSE	2	0: 5171, 1: 564
Psychiatric	0	1	FALSE	2	0: 5349, 1: 386
Pulmonary	0	1	FALSE	2	0: 4646, 1: 1089
Renal	0	1	FALSE	2	0: 5480, 1: 255
Hepatic	0	1	FALSE	2	0: 5334, 1: 401
GI.Bleed	0	1	FALSE	2	0: 5550, 1: 185
Tumor	0	1	FALSE	2	0: 4419, 1: 1316
Immunosuppression	0	1	FALSE	2	0: 4192, 1: 1543
Transfer.hx	0	1	FALSE	2	0: 5073, 1: 662
MI	0	1	FALSE	2	0: 5535, 1: 200
age	0	1	FALSE	5	[-I: 1424, [60: 1389, [70: 1338, [50: 917
sex	0	1	FALSE	2	Mal: 3192, Fem: 2543
DNR.status	0	1	FALSE	2	No: 5081, Yes: 654
Medical.insurance	0	1	FALSE	6	Pri: 1698, Med: 1458, Pri: 1236, Med: 647
Respiratory.Diag	0	1	FALSE	2	No: 3622, Yes: 2113

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
Cardiovascular.Diag	0	1	FALSE	2	No: 3804, Yes: 1931
Neurological.Diag	0	1	FALSE	2	No: 5042, Yes: 693
Gastrointestinal.Diag	0	1	FALSE	2	No: 4793, Yes: 942
Renal.Diag	0	1	FALSE	2	No: 5440, Yes: 295
Metabolic.Diag	0	1	FALSE	2	No: 5470, Yes: 265
Hematologic.Diag	0	1	FALSE	2	No: 5381, Yes: 354
Sepsis.Diag	0	1	FALSE	2	No: 4704, Yes: 1031
Trauma.Diag	0	1	FALSE	2	No: 5683, Yes: 52
Orthopedic.Diag	0	1	FALSE	2	No: 5728, Yes: 7
race	0	1	FALSE	3	whi: 4460, bla: 920, oth: 355
income	0	1	FALSE	4	Und: 3226, $11: 1165, $25: 893, > $: 451

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
edu	0	1	11.68	3.15	0.00	10.00	12.00	13.00	30.00	
DASIndex	0	1	20.50	5.32	11.00	16.06	19.75	23.43	33.00	
APACHE.score	0	1	54.67	19.96	3.00	41.00	54.00	67.00	147.00	
Glasgow.Coma.Score	0	1	21.00	30.27	0.00	0.00	0.00	41.00	100.00	
blood.pressure	0	1	78.52	38.05	0.00	50.00	63.00	115.00	259.00	
WBC	0	1	15.65	11.87	0.00	8.40	14.10	20.05	192.00	
Heart.rate	0	1	115.18	41.24	0.00	97.00	124.00	141.00	250.00	
Respiratory.rate	0	1	28.09	14.08	0.00	14.00	30.00	38.00	100.00	
Temperature	0	1	37.62	1.77	27.00	36.09	38.09	39.00	43.00	
PaO2vs.FIO2	0	1	222.27	114.95	51.60	133.32	202.50	316.69	537.50	
Albumin	0	1	3.09	0.78	0.30	2.60	3.50	3.50	29.00	
Hematocrit	0	1	31.87	8.36	2.00	26.10	30.00	36.30	66.19	
Bilirubin	0	1	2.27	4.80	0.10	0.80	1.01	1.40	58.20	
Creatinine	0	1	2.13	2.05	0.10	1.00	1.50	2.40	25.10	
Sodium	0	1	136.77	7.66	101.00	132.00	136.00	142.00	178.00	
Potassium	0	1	4.07	1.03	1.10	3.40	3.80	4.60	11.90	
PaCo2	0	1	38.75	13.18	1.00	31.00	37.00	42.00	156.00	
PH	0	1	7.39	0.11	6.58	7.34	7.40	7.46	7.77	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Weight	0	1	67.83	29.06	0.00	56.30	70.00	83.70	244.00	
Length.of.Stay	0	1	21.56	25.87	2.00	7.00	14.00	25.00	394.00	
RHC.use	0	1	0.38	0.49	0.00	0.00	0.00	1.00	1.00	

Predictive vs. causal models

The focus of current document is predictive models (e.g., predicting a health outcome).

The original article by Connors et al. (1996) focused on the association of

Connors et al. (1996)

- right heart catheterization *(RHC)* use during the first 24 hours of care in the intensive care unit (exposure of primary interest) and
- the health-outcomes (such as length of stay).

Then the PICOT table changes as follows:

Aspect	Description
P	Patients who are critically ill
I	Receiving a right heart catheterization (RHC)
C	Not receiving a right heart catheterization (RHC)
O	length of stay
T	Between 1989 to 1994 (see the JAMA paper)

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

References

Predictive question-2a

Working with a Predictive question using NHANES

Part 1: Identify, download and merge necessary data:

The tutorial focuses on building a predictive model for Diastolic blood pressure in the U.S. population for the years 2013-14. It provides a step-by-step guide on how to use R for data manipulation and analysis, covering the initial setup of the R environment, identification of relevant covariates like age, sex, and lifestyle factors, and methods to search and import these variables from the NHANES dataset. Following data importation, subsets of relevant variables are merged into a single analytic dataset, which is then saved for future research. The tutorial also includes an exercise.

Example article

Let us use the article by Li et al. (2020) as our reference. DOI:10.1038/s41371-019-0224-9.

(Li et al. 2020)

Video content (optional)

> Tip
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the content.

Research question

Building a predictive model for `Diastolic blood pressure` from 2013-14 NHANES data. Here we are only interested about explaining the outcome (`Diastolic blood pressure`).

PICOT element	Description
P	US
I	Not applicable, as we are dealing with a prediction model here
C	Not applicable, as we are dealing with a prediction model here
O	Diastolic blood pressure
T	2013-14

Covariates under consideration (that are known to influence this outcome) based on the literature (e.g., try this paper; see Table 1):

- sex
- age
- race
- marital status
- Systolic blood pressure
- smoking
- alcohol

We will also extract additional survey features: - weights, - strata, - cluster

But how do we know where to search these variables?

Searching for useful variables and datasets

R can do some preliminary searches:

Find suffix for the year

First, we need to identify the correct suffix for the data year.

`nhanesTables` function enables quick display of all available tables in survey groups:

- DEMOGRAPHICS: `DEMO`
- DIETARY: `DIET`
- EXAMINATION: `EXAM`
- LABORATORY: `LAB`
- QUESTIONNAIRE: `Q`

```
# ?nhanesTables
nhanesTables(data_group='DEMO', year=2013)
#>   Data.File.Name                    Data.File.Description
#> 1         DEMO_H Demographic Variables and Sample Weights
# nicer tables
kable(nhanesTables(data_group='DEMO', year=2013))
```

Data.File.Name	Data.File.Description
DEMO_H	Demographic Variables and Sample Weights

`H` is assigned for 2013-2014.

```
kable(nhanesTables(data_group='EXAM', year=2013))
```

Data.File.Name	Data.File.Description
BPX_H	Blood Pressure
DXXSPN_H	Dual-Energy X-ray Absorptiometry - Spine
DXXFEM_H	Dual-Energy X-ray Absorptiometry - Femur
DXXVFA_H	Dual-Energy X-ray Absorptiometry - Vertebral Fracture Assessment
BMX_H	Body Measures
DXXFRX_H	Dual-Energy X-ray Absorptiometry - FRAX Score
CSX_H	Taste & Smell
MGX_H	Muscle Strength - Grip Test

Data.File.Name	Data.File.Description
OHXDEN_H	Oral Health - Dentition
OHXPER_H	Oral Health - Periodontal
OHXREF_H	Oral Health - Recommendation of Care
FLXCLN_H	Fluorosis - Clinical
DXXAAC_H	Dual-Energy X-ray Absorptiometry - Abdominal Aortic Calcification
DXXT4_H	Dual-Energy X-ray Absorptiometry - T4 Vertebrae Morphology
DXXT5_H	Dual-Energy X-ray Absorptiometry - T5 Vertebrae Morphology
DXXT6_H	Dual-Energy X-ray Absorptiometry - T6 Vertebrae Morphology
DXXT7_H	Dual-Energy X-ray Absorptiometry - T7 Vertebrae Morphology
DXXT8_H	Dual-Energy X-ray Absorptiometry - T8 Vertebrae Morphology
DXXT9_H	Dual-Energy X-ray Absorptiometry - T9 Vertebrae Morphology
DXXT10_H	Dual-Energy X-ray Absorptiometry - T10 Vertebrae Morphology
DXXT11_H	Dual-Energy X-ray Absorptiometry - T11 Vertebrae Morphology
DXXT12_H	Dual-Energy X-ray Absorptiometry - T12 Vertebrae Morphology
DXXL1_H	Dual-Energy X-ray Absorptiometry - L1 Vertebrae Morphology
DXXL2_H	Dual-Energy X-ray Absorptiometry - L2 Vertebrae Morphology
DXXL3_H	Dual-Energy X-ray Absorptiometry - L3 Vertebrae Morphology
DXXL4_H	Dual-Energy X-ray Absorptiometry - L4 Vertebrae Morphology
DXX_H	Dual-Energy X-ray Absorptiometry - Whole Body
PAXDAY_H	Physical Activity Monitor - Day
PAXHD_H	Physical Activity Monitor - Header
PAXHR_H	Physical Activity Monitor - Hour
PAXMIN_H	Physical Activity Monitor - Minute

Data.File.Name	Data.File.Description
DXXAG_H	Dual-Energy X-ray Absorptiometry - Android/Gynoid Measurements

```
# Also try other datasets
# nhanesTables(data_group='DIET', year=2013)
```

```
# nhanesTables(data_group='LAB', year=2013)
```

```
# nhanesTables(data_group='Q', year=2013)
# what happens when you change year to 2014 or 2015.
# Try both and compare the outcome.
```

Look up variable names

Once we have the table names, we need to find out which variables in those tables are useful for us.

NHANES Variable Keyword Search

`nhanesTableVars` enables quick display of table variables and their definitions:

```
# ?nhanesTableVars
nhanesTableVars(data_group='DEMO', nh_table='DEMO_H',
                namesonly = TRUE)
#>  [1] "AIALANGA" "DMDBORN4" "DMDCITZN" "DMDEDUC2" "DMDEDUC3" "DMDFMSIZ"
#>  [7] "DMDHHSIZ" "DMDHHSZA" "DMDHHSZB" "DMDHHSZE" "DMDHRAGE" "DMDHRBR4"
#> [13] "DMDHREDU" "DMDHRGND" "DMDHRMAR" "DMDHSEDU" "DMDMARTL" "DMDYRSUS"
#> [19] "DMQADFC"  "DMQMILIZ" "FIAINTRP" "FIALANG"  "FIAPROXY" "INDFMIN2"
#> [25] "INDFMPIR" "INDHHIN2" "MIAINTRP" "MIALANG"  "MIAPROXY" "RIAGENDR"
#> [31] "RIDAGEMN" "RIDAGEYR" "RIDEXAGM" "RIDEXMON" "RIDEXPRG" "RIDRETH1"
#> [37] "RIDRETH3" "RIDSTATR" "SDDSRVYR" "SDMVPSU"  "SDMVSTRA" "SEQN"
#> [43] "SIAINTRP" "SIALANG"  "SIAPROXY" "WTINT2YR" "WTMEC2YR"
kable(nhanesTableVars(data_group='DEMO', nh_table='DEMO_H',
                      namesonly = FALSE))
```

Variable.Name	Variable.Description
AIALANGUA	Language of the MEC ACASI Interview Instrument
DMDBORN4	In what country {were you/was SP} born?
DMDCITZN	{Are you/Is SP} a citizen of the United States? [Information about citizenship is being collected by
DMDEDUC2	What is the highest grade or level of school {you have/SP has} completed or the highest degree {you
DMDEDUC3	What is the highest grade or level of school {you have/SP has} completed or the highest degree {you
DMDFMSIZ	Total number of people in the Family
DMDHHSIZ	Total number of people in the Household
DMDHHSZA	Number of children aged 5 years or younger in the household
DMDHHSZB	Number of children aged 6-17 years old in the household
DMDHHSZE	Number of adults aged 60 years or older in the household
DMDHRAGE	Reference person's age in years
DMDHRBR4	Reference person's country of birth
DMDHREDU	Reference person's education level
DMDHRGND	Reference person's gender
DMDHRMAR	Reference person's marital status
DMDHSEDU	Reference person's spouse's education level
DMDMARTL	Marital status
DMDYRSUS	Length of time the participant has been in the US.
DMQADFC	Did {you/SP} ever serve in a foreign country during a time of armed conflict or on a humanitarian or
DMQMILIZ	{Have you/Has SP} ever served on active duty in the U.S. Armed Forces, military Reserves, or Nationa
FIAINTRP	Was an interpreter used to conduct the Family interview?
FIALANGU	Language of the Family Interview Instrument
FIAPROXY	Was a Proxy respondent used in conducting the Family Interview?
INDFMIN2	Total family income (reported as a range value in dollars)
INDFMPIR	A ratio of family income to poverty guidelines.
INDHHIN2	Total household income (reported as a range value in dollars)

Variable.Name	Variable.Description
MIAINTRP	Was an interpreter used to conduct the MEC CAPI interview?
MIALANG	Language of the MEC CAPI Interview Instrument
MIAPROXY	Was a Proxy respondent used in conducting the MEC CAPI Interview?
RIAGENDR	Gender of the participant.
RIDAGEMN	Age in months of the participant at the time of screening. Reported for persons aged 24 months or yo
RIDAGEYR	Age in years of the participant at the time of screening. Individuals 80 and over are topcoded at 80
RIDEXAGM	Age in months of the participant at the time of examination. Reported for persons aged 19 years or y
RIDEXMON	Six month time period when the examination was performed - two categories: November 1 through April
RIDEXPRG	Pregnancy status for females between 20 and 44 years of age at the time of MEC exam.
RIDRETH1	Recode of reported race and Hispanic origin information
RIDRETH3	Recode of reported race and Hispanic origin information, with Non-Hispanic Asian Category
RIDSTATR	Interview and examination status of the participant.
SDDSRVYR	Data release cycle
SDMVPSU	Masked variance unit pseudo-PSU variable for variance estimation
SDMVSTRA	Masked variance unit pseudo-stratum variable for variance estimation
SEQN	Respondent sequence number.
SIAINTRP	Was an interpreter used to conduct the Sample Person (SP) interview?
SIALANG	Language of the Sample Person Interview Instrument
SIAPROXY	Was a Proxy respondent used in conducting the Sample Person (SP) interview?
WTINT2YR	Full sample 2 year interview weight.
WTMEC2YR	Full sample 2 year MEC exam weight.

```r
# https://wwwn.cdc.gov/nchs/nhanes/2013-2014/DEMO_H.htm
```

NHANES 2013-2014 Demographics Data

Displays a list of variables in the specified NHANES table:

```r
nhanesTableVars(data_group='EXAM', nh_table='BPX_H')
#>    Variable.Name                                        Variable.Description
#> 1         BPAARM                                                Arm selected:
#> 2         BPACSZ                                Cuff size (cm) (width X length)
#> 3         BPAEN1                                 Enhancement used first reading
#> 4         BPAEN2                                Enhancement used second reading
#> 5         BPAEN3                                 Enhancement used third reading
#> 6         BPAEN4                                Enhancement used fourth reading
#> 7         BPXCHR                                      60 sec HR (30 sec HR * 2)
#> 8         BPXDI1                Diastolic:  Blood pressure (first reading) mm Hg
#> 9         BPXDI2                Diastolic: Blood pressure (second reading) mm Hg
#> 10        BPXDI3                 Diastolic: Blood pressure (third reading) mm Hg
#> 11        BPXDI4 Diastolic: Blood pressure (fourth reading if necessary) mm Hg
#> 12        BPXML1                          MIL: maximum inflation levels (mm Hg)
#> 13        BPXPLS                              60 sec. pulse (30 sec. pulse * 2)
#> 14        BPXPTY                                                     Pulse type
#> 15       BPXPULS                                   Pulse regular or irregular?
#> 16        BPXSY1                 Systolic:  Blood pressure (first reading) mm Hg
#> 17        BPXSY2                Systolic:  Blood pressure (second reading) mm Hg
#> 18        BPXSY3                 Systolic:  Blood pressure (third reading) mm Hg
#> 19        BPXSY4 Systolic:  Blood pressure (fourth reading if necessary) mm Hg
#> 20      PEASCCT1                                         Blood Pressure Comment
#> 21      PEASCST1                                          Blood Pressure Status
#> 22      PEASCTM1                                 Blood Pressure Time in Seconds
#> 23          SEQN                                    Respondent sequence number.
# https://wwwn.cdc.gov/nchs/nhanes/2013-2014/BPX_H.htm
```

NHANES 2013-2014 Examination Data

Importing and Subsetting the dataset

Objective is to retain only the useful variables. We will start by importing only the demographic variables that we need.

Demographics

NHANES 2013-2014 Demographic Variables and Sample Weights (DEMO_H)

Take a look at `Target` for each variables.

What is the difference between

- `WTINT2YR`: 0 missing
- `WTMEC2YR`: Missing 0, Not MEC Examined 362

```
demo <- nhanes('DEMO_H') # Both males and females 0 YEARS - 150 YEARS
names(demo)
#>  [1] "SEQN"     "SDDSRVYR" "RIDSTATR" "RIAGENDR" "RIDAGEYR" "RIDAGEMN"
#>  [7] "RIDRETH1" "RIDRETH3" "RIDEXMON" "RIDEXAGM" "DMQMILIZ" "DMQADFC"
#> [13] "DMDBORN4" "DMDCITZN" "DMDYRSUS" "DMDEDUC3" "DMDEDUC2" "DMDMARTL"
#> [19] "RIDEXPRG" "SIALANG"  "SIAPROXY" "SIAINTRP" "FIALANG"  "FIAPROXY"
#> [25] "FIAINTRP" "MIALANG"  "MIAPROXY" "MIAINTRP" "AIALANGA" "DMDHHSIZ"
#> [31] "DMDFMSIZ" "DMDHHSZA" "DMDHHSZB" "DMDHHSZE" "DMDHRGND" "DMDHRAGE"
#> [37] "DMDHRBR4" "DMDHREDU" "DMDHRMAR" "DMDHSEDU" "WTINT2YR" "WTMEC2YR"
#> [43] "SDMVPSU"  "SDMVSTRA" "INDHHIN2" "INDFMIN2" "INDFMPIR"
demo1 <- demo[c("SEQN", # Respondent sequence number
                "RIAGENDR", # gender
                "RIDAGEYR", # Age in years at screening
                "RIDRETH3", # Race/Hispanic origin w/ NH Asian
                "DMDMARTL", # Marital status: 20 YEARS - 150 YEARS
                "WTINT2YR", "WTMEC2YR", #  Full sample 2 year weights
                "SDMVPSU", # Masked variance pseudo-PSU
                "SDMVSTRA")] # Masked variance pseudo-stratum
demo_vars <- names(demo1)
demo2 <- nhanesTranslate('DEMO_H', demo_vars, data=demo1)
#> No translation table is available for SEQN
#> Translated columns: RIAGENDR RIDRETH3 DMDMARTL
```

```
head(demo2$SEQN)
#> [1] 73557 73558 73559 73560 73561 73562
head(demo2)
#>    SEQN RIAGENDR RIDAGEYR           RIDRETH3  DMDMARTL WTINT2YR WTMEC2YR
#> 1 73557     Male       69 Non-Hispanic Black Separated 13281.24 13481.04
#> 2 73558     Male       54 Non-Hispanic White   Married 23682.06 24471.77
#> 3 73559     Male       72 Non-Hispanic White   Married 57214.80 57193.29
```

```
#> 4 73560     Male        9 Non-Hispanic White      <NA> 55201.18 55766.51
#> 5 73561   Female       73 Non-Hispanic White   Married 63709.67 65541.87
#> 6 73562     Male       56   Mexican American  Divorced 24978.14 25344.99
#>   SDMVPSU SDMVSTRA
#> 1       1      112
#> 2       1      108
#> 3       1      109
#> 4       2      109
#> 5       2      116
#> 6       1      111
```

Blood pressure

Next, we focus on the blood pressure readings.

NHANES 2013-2014 Blood Pressure (BPX_H)

Take a look at `Target` and missing for each variables. For example,

`BPXDI1` - Diastolic: Blood pres (1st rdg) mm Hg - Target:Both males and females 8 YEARS - 150 YEARS - Missing 2641

`BPXSY1` - Systolic: Blood pres (1st rdg) mm Hg - Target:Both males and females 8 YEARS - 150 YEARS - Missing 2641

```
bpx <- nhanes('BPX_H')
names(bpx)
#>  [1] "SEQN"     "PEASCST1" "PEASCTM1" "PEASCCT1" "BPXCHR"   "BPAARM"
#>  [7] "BPACSZ"   "BPXPLS"   "BPXPULS"  "BPXPTY"   "BPXML1"   "BPXSY1"
#> [13] "BPXDI1"   "BPAEN1"   "BPXSY2"   "BPXDI2"   "BPAEN2"   "BPXSY3"
#> [19] "BPXDI3"   "BPAEN3"   "BPXSY4"   "BPXDI4"   "BPAEN4"
bpx1 <- bpx[c("SEQN", # Respondent sequence number
             "BPXDI1", # Diastolic: Blood pres (1st rdg) mm Hg
             "BPXSY1")] # Systolic: Blood pres (1st rdg) mm Hg
bpx_vars <- names(bpx1)
bpx2 <- nhanesTranslate('BPX_H', bpx_vars, data=bpx1)
#> No translation table is available for SEQN
#> Warning in nhanesTranslate("BPX_H", bpx_vars, data = bpx1): No columns were
#> translated
head(bpx2)
```

```
#>    SEQN BPXDI1 BPXSY1
#> 1 73557     72    122
#> 2 73558     62    156
#> 3 73559     90    140
#> 4 73560     38    108
#> 5 73561     86    136
#> 6 73562     84    160
```

Smoking

Now, let's consider smoking data.

NHANES 2013-2014 Smoking - Cigarette Use (SMQ_H)

`SMQ040` - Do you now smoke cigarettes - Target:Both males and females 18 YEARS - 150 YEARS - Missing 4589

```
smq <- nhanes('SMQ_H')
smq1 <- smq[c("SEQN", # Respondent sequence number
              "SMQ040")] # Do you now smoke cigarettes?: 18 YEARS - 150 YEARS
smq_vars <- names(smq1)
smq2 <- nhanesTranslate('SMQ_I', smq_vars, data=smq1)
#> No translation table is available for SEQN
#> Translated columns: SMQ040
head(smq2)
#>    SEQN     SMQ040
#> 1 73557 Not at all
#> 2 73558  Some days
#> 3 73559 Not at all
#> 4 73561       <NA>
#> 5 73562 Not at all
#> 6 73564       <NA>
```

Other options for `smoking` variable candidates could be

- `SMD641` - # days smoked cigs during past 30 days
- `SMD650` - Avg # cigarettes/day during past 30 days
- `SMQ621` - Cigarettes smoked in entire life

Which of these variables are more towards describing what you are thinking as a smoking variable?

Alcohol

Finally, we will import data about alcohol consumption.

NHANES 2013-2014 Alcohol Use (ALQ_H)

`ALQ130` - Avg no alcoholic drinks/day - past 12 mos - Target: Both males and females 18 YEARS - 150 YEARS - Missing 2328

```
alq <- nhanes('ALQ_H')
alq1 <- alq[c("SEQN", # Respondent sequence number
              "ALQ130")] # Avg # alcoholic drinks/day
alq_vars <- names(alq1)
alq2 <- nhanesTranslate('ALQ_H', alq_vars, data=alq1)
#> No translation table is available for SEQN
#> Warning in nhanesTranslate("ALQ_H", alq_vars, data = alq1): No columns were
#> translated
head(alq2)
#>    SEQN ALQ130
#> 1 73557      1
#> 2 73558      4
#> 3 73559     NA
#> 4 73561     NA
#> 5 73562      1
#> 6 73564      1
```

Merging all the datasets

one-by-one

Now, we need to combine all these individual datasets into one for our analysis.

```
analytic.data0 <- merge(demo2, bpx2, by = c("SEQN"), all=TRUE)
head(analytic.data0)
#>    SEQN RIAGENDR RIDAGEYR           RIDRETH3  DMDMARTL WTINT2YR WTMEC2YR
#> 1 73557     Male       69 Non-Hispanic Black Separated 13281.24 13481.04
#> 2 73558     Male       54 Non-Hispanic White   Married 23682.06 24471.77
#> 3 73559     Male       72 Non-Hispanic White   Married 57214.80 57193.29
```

```
#> 4 73560     Male        9 Non-Hispanic White      <NA> 55201.18 55766.51
#> 5 73561   Female       73 Non-Hispanic White   Married 63709.67 65541.87
#> 6 73562     Male       56   Mexican American  Divorced 24978.14 25344.99
#>   SDMVPSU SDMVSTRA BPXDI1 BPXSY1
#> 1       1      112     72    122
#> 2       1      108     62    156
#> 3       1      109     90    140
#> 4       2      109     38    108
#> 5       2      116     86    136
#> 6       1      111     84    160
dim(analytic.data0)
#> [1] 10175    11
analytic.data1 <- merge(analytic.data0, smq2, by = c("SEQN"),
                        all=TRUE)
head(analytic.data1)
#>    SEQN RIAGENDR RIDAGEYR           RIDRETH3  DMDMARTL WTINT2YR WTMEC2YR
#> 1 73557     Male       69 Non-Hispanic Black Separated 13281.24 13481.04
#> 2 73558     Male       54 Non-Hispanic White   Married 23682.06 24471.77
#> 3 73559     Male       72 Non-Hispanic White   Married 57214.80 57193.29
#> 4 73560     Male        9 Non-Hispanic White      <NA> 55201.18 55766.51
#> 5 73561   Female       73 Non-Hispanic White   Married 63709.67 65541.87
#> 6 73562     Male       56   Mexican American  Divorced 24978.14 25344.99
#>   SDMVPSU SDMVSTRA BPXDI1 BPXSY1     SMQ040
#> 1       1      112     72    122 Not at all
#> 2       1      108     62    156  Some days
#> 3       1      109     90    140 Not at all
#> 4       2      109     38    108       <NA>
#> 5       2      116     86    136       <NA>
#> 6       1      111     84    160 Not at all
dim(analytic.data1)
#> [1] 10175    12
analytic.data2 <- merge(analytic.data1, alq2, by = c("SEQN"),
                        all=TRUE)
head(analytic.data2)
#>    SEQN RIAGENDR RIDAGEYR           RIDRETH3  DMDMARTL WTINT2YR WTMEC2YR
#> 1 73557     Male       69 Non-Hispanic Black Separated 13281.24 13481.04
#> 2 73558     Male       54 Non-Hispanic White   Married 23682.06 24471.77
#> 3 73559     Male       72 Non-Hispanic White   Married 57214.80 57193.29
#> 4 73560     Male        9 Non-Hispanic White      <NA> 55201.18 55766.51
#> 5 73561   Female       73 Non-Hispanic White   Married 63709.67 65541.87
```

```
#> 6 73562     Male       56   Mexican American  Divorced 24978.14 25344.99
#>   SDMVPSU SDMVSTRA BPXDI1 BPXSY1     SMQ040 ALQ130
#> 1       1      112     72    122 Not at all      1
#> 2       1      108     62    156  Some days      4
#> 3       1      109     90    140 Not at all     NA
#> 4       2      109     38    108       <NA>     NA
#> 5       2      116     86    136       <NA>     NA
#> 6       1      111     84    160 Not at all      1
dim(analytic.data2)
#> [1] 10175    13
```

All at once

Alternatively, you can merge all datasets at once.

```
require(plyr)
analytic.data <- join_all(list(demo2, bpx2, smq2, alq2),
                          by = "SEQN", type='full')
head(analytic.data)
#>    SEQN RIAGENDR RIDAGEYR           RIDRETH3  DMDMARTL WTINT2YR WTMEC2YR
#> 1 73557     Male       69 Non-Hispanic Black Separated 13281.24 13481.04
#> 2 73558     Male       54 Non-Hispanic White   Married 23682.06 24471.77
#> 3 73559     Male       72 Non-Hispanic White   Married 57214.80 57193.29
#> 4 73560     Male        9 Non-Hispanic White      <NA> 55201.18 55766.51
#> 5 73561   Female       73 Non-Hispanic White   Married 63709.67 65541.87
#> 6 73562     Male       56   Mexican American  Divorced 24978.14 25344.99
#>   SDMVPSU SDMVSTRA BPXDI1 BPXSY1     SMQ040 ALQ130
#> 1       1      112     72    122 Not at all      1
#> 2       1      108     62    156  Some days      4
#> 3       1      109     90    140 Not at all     NA
#> 4       2      109     38    108       <NA>     NA
#> 5       2      116     86    136       <NA>     NA
#> 6       1      111     84    160 Not at all      1
dim(analytic.data)
#> [1] 10175    13
```

Saving data for later use

It's a good practice to save your data for future reference.

```
save(analytic.data, file="Data/researchquestion/Analytic2013.RData")
```

Exercise (try yourself)

Follow the steps in the exercise section to deepen your understanding and broaden the analysis.

1. The following variables were not included in the above analysis, that were included in this paper: try including them and then create the new analytic data:

- education level
- poverty income ratio
- Sodium intake (mg)
- Potassium intake (mg)

2. Download the NHANES 2015-2016 and append with the NHANES 2013-2014 analytic data with same variables.

References

Predictive question-2b

Working with a Predictive question using NHANES

Part 2: Analysis of downloaded data:

This tutorial provides a comprehensive guide to NHANES data preparation and initial analysis using R. The tutorial covers topics such as loading dataset, variable recoding, data summary statistics, and various types of regression analyses, including bivariate and multivariate models. It also delves into dealing with missing data, first by omitting NA values for a complete case analysis and then using a simple imputation method. The guide is designed to walk the reader through each step of data manipulation and analysis, with a focus on avoiding common pitfalls in statistical analysis.

Example article

We are continuing to use the article by Li et al. (2020) as our reference. DOI:10.1038/s41371-019-0224-9.

(Li et al. 2020)

Video content (optional)

> Tip
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the content.

Loading saved data

The following code chunk loads an RData file named that was saved in the previous step. The RData file typically contains saved R objects like data frames, lists, etc.

```
load("Data/researchquestion/Analytic2013.RData")
```

The following code lists all the objects in the current workspace.

```
ls()
#> [1] "analytic.data"   "has_annotations"
```

The following shows the column names of the `analytic.data` data frame.

```
names(analytic.data)
#>  [1] "SEQN"     "RIAGENDR" "RIDAGEYR" "RIDRETH3" "DMDMARTL" "WTINT2YR"
#>  [7] "WTMEC2YR" "SDMVPSU"  "SDMVSTRA" "BPXDI1"   "BPXSY1"   "SMQ040"
#> [13] "ALQ130"
```

The following provides summary statistics for the column `BPXDI1` in the `analytic.data` data frame.

```
summary(analytic.data$BPXDI1)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>    0.00   58.00   66.00   65.77   76.00  122.00    3003
```

Target

First, we need to understand the data well. What is the common support in all of the population?

Variable	Target
SEQN	Both males and females 0 YEARS - 150 YEARS
RIAGENDR	Both males and females 0 YEARS - 150 YEARS
RIDAGEYR	Both males and females 0 YEARS - 150 YEARS

Variable	Target
RIDRETH3	Both males and females 0 YEARS - 150 YEARS
DMDMARTL	Both males and females 20 YEARS - 150 YEARS
WTINT2YR	Both males and females 0 YEARS - 150 YEARS
WTMEC2YR	Both males and females 0 YEARS - 150 YEARS
SDMVPSU	Both males and females 0 YEARS - 150 YEARS
SDMVSTRA	Both males and females 0 YEARS - 150 YEARS
BPXDI1	Both males and females 8 YEARS - 150 YEARS
BPXSY1	Both males and females 8 YEARS - 150 YEARS
SMQ040	Both males and females 18 YEARS - 150 YEARS
ALQ130	Both males and females 18 YEARS - 150 YEARS
-	-

Both males and females 20 YEARS - 150 YEARS. The study should be restricted to age 20 and +.

Recode and Univariate summary

Blood pressure

```
require(car)
summary(analytic.data$BPXSY1)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>    66.0   106.0   116.0   118.1   128.0   228.0    3003
summary(analytic.data$BPXDI1)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>    0.00   58.00   66.00   65.77   76.00  122.00    3003
# what is 0 blood pressure?
# change all 0 to NA
analytic.data$BPXDI1 <- recode(analytic.data$BPXDI1, "0=NA")
```

```
#create a new variable
analytic.data$blood.pressure <- analytic.data$BPXDI1
```

```
# delete old variable
analytic.data$BPXDI1 <- NULL
```

```r
# check summary
summary(analytic.data$blood.pressure)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>    4.00   58.00   68.00   66.54   76.00  122.00    3086
```

Race

The `RIDRETH3` column is recoded to simplify racial categories.

```r
analytic.data$RIDRETH3 <-
  recode(analytic.data$RIDRETH3,
         "c('Non-Hispanic Asian',
         'Other Race - Including Multi-Rac')='Other race'")
analytic.data$race <- analytic.data$RIDRETH3
analytic.data$RIDRETH3 <- NULL
table(analytic.data$race,useNA="always")
#>
#>   Mexican American Non-Hispanic Black Non-Hispanic White     Other Hispanic
#>               1730               2267               3674                960
#>         Other race               <NA>
#>               1544                  0
```

Age (centering)

Age values are centered around the mean age for those who are 20 years or older.

```r
summary(analytic.data$RIDAGEYR)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    0.00   10.00   26.00   31.48   52.00   80.00
centre.adult <- mean(analytic.data$RIDAGEYR[analytic.data$RIDAGEYR
                                            >= 20], na.rm = TRUE)
centre.adult
#> [1] 49.11111
# This is actually not the correct mean age. Guess why?
# Hint: see the NHANES data dictionary for age variable.
analytic.data$RIDAGEYRc <- analytic.data$RIDAGEYR - centre.adult
analytic.data$age.centred <- analytic.data$RIDAGEYRc
```

```r
analytic.data$RIDAGEYRc <- NULL
summary(analytic.data$age.centred)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> -49.111 -39.111 -23.111 -17.627   2.889  30.889
```

Gender

A new column `gender` is created for gender details.

```r
analytic.data$gender <- analytic.data$RIAGENDR
analytic.data$RIAGENDR <- NULL
table(analytic.data$gender, useNA = "always")
#>
#>   Male Female   <NA>
#>   5003   5172      0
```

Marital status

The marital status is simplified.

```r
summary(analytic.data$DMDMARTL)
#>             Married             Widowed            Divorced           Separated
#>                2965                 436                 659                 177
#>       Never married Living with partner             Refused          Don't Know
#>                1112                 417                   2                   1
#>                NA's
#>                4406
analytic.data$DMDMARTL <-
  recode(analytic.data$DMDMARTL,
  "c('Widowed','Divorced','Separated')='Previously married';
  c('Living with partner','Married')='Married';
  'Never married' = 'Never married';
  else=NA")

# what happened to 77 and 99? Hint: else
analytic.data$marital <- analytic.data$DMDMARTL
analytic.data$DMDMARTL <- NULL
table(analytic.data$marital, useNA = "always")
```

```
#>
#>            Married      Never married Previously married                <NA>
#>               3382               1112               1272               4409
```

Alcohol

For Alcohol, codes like 777 and 999 are converted to `NA`.

```
summary(analytic.data$ALQ130)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>   1.000   1.000   2.000   3.511   3.000 999.000    6579
# what is  777 and 999? See NHANES data dictionary.
# Refused and Don't know
analytic.data$ALQ130[analytic.data$ALQ130 == 999] <- NA
analytic.data$ALQ130[analytic.data$ALQ130 == 777] <- NA
analytic.data$alcohol <- analytic.data$ALQ130
analytic.data$ALQ130 <- NULL
table(analytic.data$alcohol, useNA = "always")
#>
#>    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
#> 1280 1002  536  283  154  156   23   54    4   28    4   43    5    2    6    4
#>   18   20   24   25 <NA>
#>    3    4    1    1 6582
```

Smoking

Similar to alcohol, unusual codes are converted to `NA`.

```
summary(analytic.data$SMQ040)
#>  Every day  Some days Not at all       NA's
#>        992        240       1347       7596
# what is  7 and 9? See NHANES data dictionary.
# Refused and Don't know
analytic.data$SMQ040[analytic.data$SMQ040 == 9] <- NA
analytic.data$SMQ040[analytic.data$SMQ040 == 7] <- NA
analytic.data$smoke <- analytic.data$SMQ040
analytic.data$SMQ040 <- NULL
table(analytic.data$smoke, useNA = "always")
```

```
#>
#>  Every day  Some days Not at all       <NA>
#>        992        240       1347       7596
```

Renaming

Columns in the data frame are renamed for better readability.

```
names(analytic.data)
#>  [1] "SEQN"           "RIDAGEYR"       "WTINT2YR"       "WTMEC2YR"
#>  [5] "SDMVPSU"        "SDMVSTRA"       "BPXSY1"         "blood.pressure"
#>  [9] "race"           "age.centred"    "gender"         "marital"
#> [13] "alcohol"        "smoke"
```

Order is important.

```
names(analytic.data) <- c("id", "age", "w.all", "w.MEC",
                          "PSU", "STRATA", "systolic",
                          "diastolic", "race", "age.centred",
                          "gender", "marital", "alcohol",
                          "smoke")
```

```
names(analytic.data)
#>  [1] "id"          "age"         "w.all"       "w.MEC"       "PSU"
#>  [6] "STRATA"      "systolic"    "diastolic"   "race"        "age.centred"
#> [11] "gender"      "marital"     "alcohol"     "smoke"
```

Subsetting to avoid zero-cells

A new age category (`age.cat`) is created to segregate the data. Remember that, the target for marital status component was `20 YEARS - 150 YEARS`.

```
summary(analytic.data$age)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    0.00   10.00   26.00   31.48   52.00   80.00
```

```r
analytic.data$age.cat <- cut(analytic.data$age,
                             breaks = c(-Inf, 20,50,+Inf),
                             right = FALSE)
```

```r
table(analytic.data$age.cat)
#>
#> [-Inf,20)   [20,50) [50, Inf)
#>      4406      2989      2780
```

```r
summary(analytic.data$alcohol)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>    1.00    1.00    2.00    2.68    3.00   25.00    6582
```

```r
table(analytic.data$age.cat, analytic.data$marital)
#>
#>              Married Never married Previously married
#>   [-Inf,20)       0             0                  0
#>   [20,50)      1771           890                327
#>   [50, Inf)    1611           222                945
```

```r
dim(analytic.data)
#> [1] 10175    15
analytic.data1 <- subset(analytic.data ,
                         !is.na(age) & age >= 20)
dim(analytic.data1)
#> [1] 5769   15
```

Note: This subsetting is problematic for further survey data analysis. We will learn to work with this subsetting correctly later.

```r
analytic.data1$age <-NULL
```

Summary

This part provides various summary statistics for the processed data.

Univariate summary for the complete case

```
summary(analytic.data1$diastolic)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>     4.0    62.0    70.0    70.3    78.0   122.0     686
```

```
table(analytic.data1$race,useNA = "always")
#>
#>   Mexican American Non-Hispanic Black Non-Hispanic White     Other Hispanic
#>                767               1177               2472                508
#>         Other race               <NA>
#>                845                  0
```

```
summary(analytic.data1$age.centred)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> -29.111 -15.111  -1.111   0.000  13.889  30.889
```

```
table(analytic.data1$gender,useNA = "always")
#>
#>   Male Female   <NA>
#>   2758   3011      0
```

```
table(analytic.data1$marital,useNA = "always")
#>
#>            Married      Never married Previously married                <NA>
#>               3382               1112               1272                   3
```

Saving for further use

```
save(analytic.data1,
     file = "Data/researchquestion/NHANESanalytic.Rdata")
```

Regression summary (Optional)

This is optional content for this chapter. Later in confounding and predictive factor chapters, we will learn more about adjustment.

Different Generalized Linear Models (GLMs) are fit for diastolic blood pressure using variables like gender, marital status, etc. Below, we used GLMs with the Gaussian family for the continuous outcome diastolic. Note that `gaussian` is the default family for the `glm` function. Some other family options include binomial, poisson, quasibinomial, and so on.

Bivariate Regression summary (missing values included)

```
fit1g <- glm(diastolic ~ gender, data = analytic.data1)
summary(fit1g)
#>
#> Call:
#> glm(formula = diastolic ~ gender, data = analytic.data1)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -67.579   -7.091    0.421    6.909   50.421
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   71.5789     0.2352 304.299  < 2e-16 ***
#> genderFemale  -2.4880     0.3278  -7.591 3.76e-14 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 136.3911)
#>
#>     Null deviance: 700862  on 5082  degrees of freedom
#> Residual deviance: 693003  on 5081  degrees of freedom
#>   (686 observations deleted due to missingness)
#> AIC: 39415
#>
#> Number of Fisher Scoring iterations: 2
```

```
fit1m <- glm(diastolic ~ marital, data=analytic.data1)
summary(fit1m)
#>
```

```
#> Call:
#> glm(formula = diastolic ~ marital, data = analytic.data1)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -66.750   -6.838    1.162    7.250   51.250
#>
#> Coefficients:
#>                           Estimate Std. Error t value Pr(>|t|)
#> (Intercept)                70.7500     0.2138 330.901  < 2e-16 ***
#> maritalNever married       -1.9116     0.4316  -4.429 9.69e-06 ***
#> maritalPreviously married  -0.3953     0.4140  -0.955     0.34
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 137.5101)
#>
#>     Null deviance: 700840  on 5079  degrees of freedom
#> Residual deviance: 698139  on 5077  degrees of freedom
#>   (689 observations deleted due to missingness)
#> AIC: 39434
#>
#> Number of Fisher Scoring iterations: 2
```

```
str(analytic.data1)
#> 'data.frame':    5769 obs. of  14 variables:
#>  $ id         : num  73557 73558 73559 73561 73562 ...
#>  $ w.all      : num  13281 23682 57215 63710 24978 ...
#>  $ w.MEC      : num  13481 24472 57193 65542 25345 ...
#>  $ PSU        : num  1 1 1 2 1 1 2 1 2 2 ...
#>  $ STRATA     : num  112 108 109 116 111 114 106 112 112 113 ...
#>  $ systolic   : num  122 156 140 136 160 118 NA 128 140 106 ...
#>  $ diastolic  : num  72 62 90 86 84 80 NA 74 78 60 ...
#>  $ race       : Factor w/ 5 levels "Mexican American",..: 2 3 3 3 1 3 4 3 3 3 ...
#>  $ age.centred: num  19.89 4.89 22.89 23.89 6.89 ...
#>  $ gender     : Factor w/ 2 levels "Male","Female": 1 1 1 2 1 2 1 2 1 2 ...
#>  $ marital    : Factor w/ 3 levels "Married","Never married",..: 3 1 1 1 3 3 1 3 3 2 ...
#>  $ alcohol    : num  1 4 NA NA 1 1 NA 1 3 2 ...
#>  $ smoke      : Factor w/ 3 levels "Every day","Some days",..: 3 2 3 NA 3 NA 3 1 1 NA ...
#>  $ age.cat    : Factor w/ 3 levels "[-Inf,20)","[20,50)",..: 3 3 3 3 3 3 2 3 3 2 ...
```

```r
fit13 <- glm(diastolic ~ gender + age.centred + race +
               marital + systolic + smoke + alcohol,
             data = analytic.data1)
summary(fit13)
#>
#> Call:
#> glm(formula = diastolic ~ gender + age.centred + race + marital +
#>     systolic + smoke + alcohol, data = analytic.data1)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -75.142   -6.090    0.811    7.074   33.512
#>
#> Coefficients:
#>                           Estimate Std. Error t value Pr(>|t|)
#> (Intercept)               30.92372    2.44895  12.627  < 2e-16 ***
#> genderFemale              -0.34850    0.59830  -0.582 0.560325
#> age.centred               -0.13638    0.02142  -6.367 2.56e-10 ***
#> raceNon-Hispanic Black     1.44736    1.11246   1.301 0.193443
#> raceNon-Hispanic White     0.59565    0.96117   0.620 0.535540
#> raceOther Hispanic         1.07369    1.29793   0.827 0.408234
#> raceOther race             2.02908    1.22998   1.650 0.099216 .
#> maritalNever married      -2.92801    0.79123  -3.701 0.000223 ***
#> maritalPreviously married  0.44754    0.71911   0.622 0.533804
#> systolic                   0.31071    0.01763  17.624  < 2e-16 ***
#> smokeSome days            -0.42177    0.97853  -0.431 0.666513
#> smokeNot at all            0.01796    0.65159   0.028 0.978008
#> alcohol                    0.17287    0.10994   1.572 0.116060
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 117.7142)
#>
#>     Null deviance: 219477  on 1515  degrees of freedom
#> Residual deviance: 176924  on 1503  degrees of freedom
#>   (4253 observations deleted due to missingness)
#> AIC: 11546
#>
#> Number of Fisher Scoring iterations: 2
```

Check missingness (optional)

The `plot_missing()` function from the `DataExplorer` package is used to plot missing data.

A subsequent chapter will delve into the additional factors that impact how we handle missing data.

```
require(DataExplorer)
plot_missing(analytic.data1)
```

```
require("tableone")
vars = c("systolic", "smoke", "diastolic", "race",
         "age.centred", "gender", "marital", "alcohol")
CreateTableOne(data = analytic.data1, includeNA = TRUE,
               vars = vars)
#>
#>                            Overall
#>   n                          5769
#>   systolic (mean (SD))    123.16 (18.12)
#>   smoke (%)
#>      Every day               965 (16.7)
#>      Some days               229 ( 4.0)
#>      Not at all             1336 (23.2)
#>      NA                     3239 (56.1)
#>   diastolic (mean (SD))    70.30 (11.74)
#>   race (%)
```

```
#>      Mexican American        767 (13.3)
#>      Non-Hispanic Black     1177 (20.4)
#>      Non-Hispanic White     2472 (42.8)
#>      Other Hispanic          508 ( 8.8)
#>      Other race              845 (14.6)
#>   age.centred (mean (SD))   0.00 (17.56)
#>   gender = Female (%)       3011 (52.2)
#>   marital (%)
#>      Married                3382 (58.6)
#>      Never married          1112 (19.3)
#>      Previously married     1272 (22.0)
#>      NA                        3 ( 0.1)
#>   alcohol (mean (SD))       2.65 (2.34)
```

Setting correct variable types

The variables are explicitly set to either numeric or factor types.

Note: In case any of the variables types are wrong, your table 1 output will be wrong. Better to be sure about what type of variable you want them to be (numeric or factor). For example, `systolic` should be numeric. Is it defined that way?

```
mode(analytic.data1$systolic)
#> [1] "numeric"
```

In case it wasn't (often they can get converted to character), then here is the solution:

```
# solution 1: one-by-one
analytic.data1$systolic <-
  as.numeric(as.character(analytic.data1$systolic))
summary(analytic.data1$systolic)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>    66.0   110.0   120.0   123.2   134.0   228.0     658
```

```r
# solution 2: fixing all variable types at once
numeric.names <- c("systolic", "diastolic",
                   "age.centred", "alcohol")
factor.names <- vars[!vars %in% numeric.names]
factor.names
#> [1] "smoke"   "race"    "gender"  "marital"
analytic.data1[,factor.names] <-
  lapply(analytic.data1[,factor.names] , factor)
analytic.data1[numeric.names] <-
  apply(X = analytic.data1[numeric.names], MARGIN = 2,
        FUN =function (x) as.numeric(as.character(x)))
levels(analytic.data1$marital)
#> [1] "Married"            "Never married"      "Previously married"
CreateTableOne(data = analytic.data1, includeNA = TRUE,
               vars = vars)
#>
#>                            Overall
#>   n                          5769
#>   systolic (mean (SD))    123.16 (18.12)
#>   smoke (%)
#>      Every day               965 (16.7)
#>      Some days               229 ( 4.0)
#>      Not at all             1336 (23.2)
#>      NA                     3239 (56.1)
#>   diastolic (mean (SD))    70.30 (11.74)
#>   race (%)
#>      Mexican American        767 (13.3)
#>      Non-Hispanic Black     1177 (20.4)
#>      Non-Hispanic White     2472 (42.8)
#>      Other Hispanic          508 ( 8.8)
#>      Other race              845 (14.6)
#>   age.centred (mean (SD))   0.00 (17.56)
#>   gender = Female (%)       3011 (52.2)
#>   marital (%)
#>      Married                3382 (58.6)
#>      Never married          1112 (19.3)
#>      Previously married     1272 (22.0)
#>      NA                        3 ( 0.1)
#>   alcohol (mean (SD))       2.65 (2.34)
```

Complete case analysis

Removes all rows containing `NA`.

```
dim(analytic.data1)
#> [1] 5769   14
analytic.data2 <- as.data.frame(na.omit(analytic.data1))
dim(analytic.data2)
#> [1] 1516   14
plot_missing(analytic.data2)
```

```
tab1 <- CreateTableOne(data = analytic.data2, includeNA = TRUE,
               vars = vars)
print(tab1)
#>
#>                           Overall
#>   n                         1516
#>   systolic (mean (SD))    123.29 (17.58)
#>   smoke (%)
#>      Every day              590 (38.9)
#>      Some days              159 (10.5)
#>      Not at all             767 (50.6)
#>   diastolic (mean (SD))    70.11 (12.04)
#>   race (%)
#>      Mexican American       162 (10.7)
```

```
#>      Non-Hispanic Black      292 (19.3)
#>      Non-Hispanic White      778 (51.3)
#>      Other Hispanic          126 ( 8.3)
#>      Other race              158 (10.4)
#>   age.centred (mean (SD))  -0.76 (16.71)
#>   gender = Female (%)        626 (41.3)
#>   marital (%)
#>      Married                 858 (56.6)
#>      Never married           300 (19.8)
#>      Previously married      358 (23.6)
#>   alcohol (mean (SD))       3.15 (2.76)
# For categorical variables, try to see if
# any categories have 0% or 100% frequency.
# If yes, those may create problem in further analysis.
```

We can export the tableone to a csv file as follows:

```
tabl1p <- print(tab1)
#>
#>                            Overall
#>   n                          1516
#>   systolic (mean (SD))    123.29 (17.58)
#>   smoke (%)
#>      Every day               590 (38.9)
#>      Some days               159 (10.5)
#>      Not at all              767 (50.6)
#>   diastolic (mean (SD))    70.11 (12.04)
#>   race (%)
#>      Mexican American        162 (10.7)
#>      Non-Hispanic Black      292 (19.3)
#>      Non-Hispanic White      778 (51.3)
#>      Other Hispanic          126 ( 8.3)
#>      Other race              158 (10.4)
#>   age.centred (mean (SD))  -0.76 (16.71)
#>   gender = Female (%)        626 (41.3)
#>   marital (%)
#>      Married                 858 (56.6)
#>      Never married           300 (19.8)
#>      Previously married      358 (23.6)
```

```
#>   alcohol (mean (SD))       3.15 (2.76)
write.csv(tabl1p, file = "Data/researchquestion/table1.csv")
```

```
fit23 <- glm(diastolic ~ gender + age.centred + race +
               marital + systolic + smoke + alcohol,
             data = analytic.data2)
require(Publish)
publish(fit23)
#>     Variable              Units Coefficient         CI.95     p-value
#>  (Intercept)                          30.92 [26.12;35.72]     < 1e-04
#>       gender                Male         Ref
#>                            Female       -0.35  [-1.52;0.82]   0.5603254
#>  age.centred                          -0.14 [-0.18;-0.09]     < 1e-04
#>         race   Mexican American         Ref
#>              Non-Hispanic Black        1.45  [-0.73;3.63]   0.1934428
#>              Non-Hispanic White        0.60  [-1.29;2.48]   0.5355396
#>                  Other Hispanic        1.07  [-1.47;3.62]   0.4082336
#>                      Other race        2.03  [-0.38;4.44]   0.0992165
#>      marital            Married         Ref
#>                   Never married       -2.93 [-4.48;-1.38]   0.0002229
#>              Previously married        0.45  [-0.96;1.86]   0.5338035
#>     systolic                           0.31   [0.28;0.35]     < 1e-04
#>        smoke          Every day         Ref
#>                       Some days       -0.42  [-2.34;1.50]   0.6665127
#>                      Not at all        0.02  [-1.26;1.30]   0.9780080
#>      alcohol                           0.17  [-0.04;0.39]   0.1160603
```

Imputed data

We will learn about proper missing data analysis at a latter class. Currently, we will do a simple (but rather controversial) single imputation. In here we are simply using a random sampling to impute (probably the worst method, but we are just filling in some gaps for now).

```
require(mice)
imputation1 <- mice(analytic.data1,
                    method = "sample",
                    m = 1, # Number of multiple imputations
```

```
                    maxit = 1 # #iteration; mostly useful for convergence
                    )
#>
#>  iter imp variable
#>   1   1  systolic  diastolic  marital  alcohol  smoke
#> Warning: Number of logged events: 5
analytic.data.imputation1 <- complete(imputation1)
dim(analytic.data.imputation1)
#> [1] 5769   14
str(analytic.data.imputation1)
#> 'data.frame':    5769 obs. of  14 variables:
#>  $ id         : num  73557 73558 73559 73561 73562 ...
#>  $ w.all      : num  13281 23682 57215 63710 24978 ...
#>  $ w.MEC      : num  13481 24472 57193 65542 25345 ...
#>  $ PSU        : num  1 1 1 2 1 1 2 1 2 2 ...
#>  $ STRATA     : num  112 108 109 116 111 114 106 112 112 113 ...
#>  $ systolic   : num  122 156 140 136 160 118 156 128 140 106 ...
#>  $ diastolic  : num  72 62 90 86 84 80 60 74 78 60 ...
#>  $ race       : Factor w/ 5 levels "Mexican American",..: 2 3 3 3 1 3 4 3 3 3 ...
#>  $ age.centred: num  19.89 4.89 22.89 23.89 6.89 ...
#>  $ gender     : Factor w/ 2 levels "Male","Female": 1 1 1 2 1 2 1 2 1 2 ...
#>  $ marital    : Factor w/ 3 levels "Married","Never married",..: 3 1 1 1 3 3 1 3 3 2 ...
#>  $ alcohol    : num  1 4 1 1 1 1 3 1 3 2 ...
#>  $ smoke      : Factor w/ 3 levels "Every day","Some days",..: 3 2 3 3 3 3 3 1 1 3 ...
#>  $ age.cat    : Factor w/ 3 levels "[-Inf,20)","[20,50)",..: 3 3 3 3 3 3 2 3 3 2 ...
plot_missing(analytic.data.imputation1)
```

```
CreateTableOne(data = analytic.data.imputation1, includeNA = TRUE,
               vars = vars)
#>
#>                           Overall
#>   n                         5769
#>   systolic (mean (SD))    123.09 (18.03)
#>   smoke (%)
#>      Every day              2264 (39.2)
#>      Some days               502 ( 8.7)
#>      Not at all             3003 (52.1)
#>   diastolic (mean (SD))    70.25 (11.73)
#>   race (%)
#>      Mexican American        767 (13.3)
#>      Non-Hispanic Black     1177 (20.4)
#>      Non-Hispanic White     2472 (42.8)
#>      Other Hispanic          508 ( 8.8)
#>      Other race              845 (14.6)
#>   age.centred (mean (SD))   0.00 (17.56)
#>   gender = Female (%)       3011 (52.2)
#>   marital (%)
#>      Married                3384 (58.7)
#>      Never married          1113 (19.3)
#>      Previously married     1272 (22.0)
#>   alcohol (mean (SD))       2.70 (2.45)
# For categorical variables, try to see if
```

```r
# any categories have 0% or 100% frequency.
# If yes, those may create problem in further analysis.
```

```r
fit23i <- glm(diastolic ~ gender + age.centred + race +
                marital + systolic + smoke + alcohol,
              data = analytic.data.imputation1)
publish(fit23i)
#>     Variable              Units Coefficient         CI.95    p-value
#>  (Intercept)                         39.61 [37.24;41.98]    < 1e-04
#>       gender               Male         Ref
#>                          Female       -1.40 [-1.98;-0.83]    < 1e-04
#>  age.centred                         -0.12 [-0.14;-0.10]    < 1e-04
#>         race   Mexican American         Ref
#>              Non-Hispanic Black        0.89  [-0.12;1.90]   0.083709
#>              Non-Hispanic White        0.49  [-0.40;1.38]   0.277214
#>                  Other Hispanic        0.38  [-0.84;1.60]   0.544173
#>                      Other race        1.58   [0.51;2.65]   0.003694
#>      marital            Married         Ref
#>                   Never married       -2.63 [-3.43;-1.83]    < 1e-04
#>              Previously married       -0.21  [-0.95;0.53]   0.575612
#>     systolic                           0.26   [0.24;0.27]    < 1e-04
#>        smoke          Every day         Ref
#>                       Some days        0.01  [-1.05;1.07]   0.982913
#>                      Not at all       -0.31  [-0.92;0.29]   0.307166
#>      alcohol                           0.04  [-0.09;0.16]   0.556142
```

We see some changes in the estimates. After imputing compared to complete case analysis, any changes dramatic (e.g., changing conclusion)?

```r
require(jtools)
require(ggstance)
require(broom.mixed)
require(huxtable)
export_summs(fit23, fit23i)
```

Additional factors come into play when dealing with complex survey datasets; these will be explored in a subsequent chapter.

```r
plot_summs(fit23, fit23i)
```

```r
# plot_summs(fit23, fit23i, plot.distributions = TRUE)
```

Exercise (try yourself)

In this lab, we have done multiple steps that could be improved. One of them was single imputation by random sampling. What other ad hoc method you could use to impute the factor variables?

References

	Model 1	Model 2
(Intercept)	30.92 ***	39.34 ***
	(2.45)	(1.21)
genderFemale	-0.35	-1.20 ***
	(0.60)	(0.30)
age.centred	-0.14 ***	-0.11 ***
	(0.02)	(0.01)
raceNon-Hispanic Black	1.45	1.15 *
	(1.11)	(0.51)
raceNon-Hispanic White	0.60	0.64
	(0.96)	(0.45)
raceOther Hispanic	1.07	0.42
	(1.30)	(0.62)
raceOther race	2.03	1.98 ***
	(1.23)	(0.54)
maritalNever married	-2.93 ***	-2.44 ***
	(0.79)	(0.41)
maritalPreviously married	0.45	-0.02
	(0.72)	(0.38)
systolic	0.31 ***	0.25 ***
	(0.02)	(0.01)
smokeSome days	-0.42	-0.16
	(0.98)	(0.54)
smokeNot at all	0.02	-0.16
	(0.65)	(0.31)
alcohol	0.17	-0.01
	237 (0.11)	(0.06)
N	1516	5769
AIC	11545.85	43925.63
BIC	11620.38	44018.87
Pseudo R2	0.19	0.14

Causal question-1

Working with a Predictive question using CCHS

Load data

We download and process the data in the same was as shown earlier, and reuse the data for the following tutorial.

```
load(file = "Data/researchquestion/cchsc1.RData")
load(file = "Data/researchquestion/cchsc2.RData")
load(file = "Data/researchquestion/cchsc3.RData")
ls() # see list of objects available
#> [1] "c1"              "c2"              "c3"                "has_annotations"
#> [5] "use.saved.chche"
dim(c1) # Dimensions of CCHS 1.1
#> [1] 130880    117
dim(c2) # Dimensions of CCHS 2.1
#> [1] 134072    112
dim(c3) # Dimensions of CCHS 3.1
#> [1] 132221    112
```

Aim

Rheumatic diseases may cause acute or chronic inflammation. Such systemic inflammation could increase the risk of cardiovascular diseases (CVD). However, Osteoarthritis (OA) is not so well explored in association with CVD. The aim of the current study is to examine the association between OA and CVD among Canadian adults.

Example article

We are going to use the article by Rahman et al. (2013) as our reference. DOI:10.1136/bmjopen-2013-002624.

(Rahman et al. 2013)

We will revisit this question and data again in the survey data chapter.

PICOT

- Target population: Canadian adults
- Outcome (Y): CVD (Heart disease)
- Exposure group (A): OA
- Control group: People without OA
- Time line: From 2001 - 2005
- From the literature, we have identified some factors to be useful in exploring this $A - Y$ relationship:

```
- age,
- sex,
- income,
- ethnicity,
- obesity,
- exercise,
- smoking,
- diets (Proxied by fruit and vegetable consumption),
- pain medication use,
- hypertension,
- cholesterol (Proxied by Diabetes and Chronic obstructive pulmonary disease /COPD),
- Additionally we have considered education level.
```

Why this time frame?

In CCHS cycle 1.1-3.1, there was a question 'What type of arthritis?' - which resulted in responses such as

- Rheumatoid arthritis,
- OA,
- other,

- unknown.

This question was crucial in identifying OA patients. However, in later years, that question was omitted. Hence, for this practical reason, we restricted out analysis in CCHS cycle 1.1-3.1 (2001-2005).

Creating Analytic Data

Identify Relevant factors

From the literature, we identify variables that are associated with either the outcome and/or the exposure. We have to be cautious about the variables that are only related to exposures only (a topic for later).

Generally try to include basic demographics (e.g., age, sex, education etc. are the usual suspects) at this stage even if we do not have a strong indication from the literature that those variables are highly associated with the outcome or the exposure.

Variable name	Variable	Type	Categories
OA	Exposure	Binary	
CVD	Outcome	Binary	
Age	Covariate	Category	20-39; 40-49; 50-59; 60-64
Sex	Covariate	Binary	Men; Women
Income	Covariate	Category	<30k; 30k-49k; 50k-79k; 80k+
Cultural/racial origin	Covariate	Binary	White; Visible minority
BMI	Covariate	Category	Underweight; Normal; Overweight; Obese
Physical activity	Covariate	Category	Active; Moderate; Inactive
Smoking	Covariate	Category	Non-smoker; Currently; Former

Variable name	Variable	Type	Categories
Fruit/vegetable consumption	Covariate	Category	0-3; 4-6 ; 6+ servings daily
Pain medication use	Covariate	Binary	
Hypertension	Covariate	Binary	
COPD	Covariate	Binary	
Diabetes	Covariate	Binary	
Education	Covariate	Category	<Secondary; Secondary graduate; PostSecondary+; PostSecondary grad
Weight	Sampling weights	Continuous	

Variables under consideration

Find out whether we have the variables collected / measured / asked in the surveys. If not asked, try to find out whether there is a proxy variable that was collected / measured / asked in the surveys.

For example, we did not have a question / variable related to a particular topic (e.g., say, cholesterol, which is known to be associated with our outcome), try to collect some variables that could be used as proxies (e.g., COPD, Diabetes).

Note that the variable names are generally different in different cycles.

Variable name	CCHS 1.1	CCHS 2.1	CCHS 3.1
Arthritis	CCCA_051	CCCC_051	CCCE_051
Kind of arthritis (for OA)	CCCA_05A	CCCC_05A	CCCE_05A
Heart disease (CVD)	CCCA_121	CCCC_121	CCCE_121
Age	DHHAGAGE	DHHCGAGE	DHHEGAGE
Sex	DHHA_SEX	DHHC_SEX	DHHE_SEX
Household income	INCAGHH	INCCGHH	INCEGHH

Variable name	CCHS 1.1	CCHS 2.1	CCHS 3.1
Cultural/racial origin	SDCAGRAC	SDCCGRAC	SDCEGCGT
BMI (Score)	HWTAGBMI	HWTCGBMI	HWTEGBMI
BMI (Category)	HWTAGSW	HWTCGISW	HWTEGISW
Physical activity	PACADPAI	PACCDPAI	PACEDPAI
Smoking status (Type of smoker)	SMKADSTY	SMKCDSTY	SMKEDSTY
Fruits/vegetables consumption	FVCADTOT	FVCCDTOT	FVCEDTOT
Pain med use	DRGA_1A	MEDC_1A	MEDE_1A
Hypertension	CCCA_071	CCCC_071	CCCE_071
Has emphysema or COPD	CCCA_91B	CCCC_91B	CCCE_91F
Diabetes	CCCA_101	CCCC_101	CCCE_101
Education	EDUADR04	EDUCDR04	EDUEDR04
Survey weights	WTSAM	WTSC_M	WTSE_M

Subset the data

```r
# Restrict the dataset with variables of interest only
var.names1 <- c("CCCA_051", "CCCA_05A", "CCCA_121", "DHHAGAGE",
               "DHHA_SEX", "INCAGHH", "SDCAGRAC", "HWTAGBMI",
               "HWTAGSW", "PACADPAI", "SMKADSTY", "FVCADTOT",
               "DRGA_1A", "CCCA_071", "CCCA_91B", "CCCA_101",
               "EDUADR04", "WTSAM")
cc11 <- c1[var.names1]
dim(cc11)
#> [1] 130880     18
table(cc11$CCCA_051)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          24511         106231              0            110              3
#>     NOT STATED
#>             25
var.names2 <- c("CCCC_051", "CCCC_05A", "CCCC_121", "DHHCGAGE",
               "DHHC_SEX", "INCCGHH", "SDCCGRAC", "HWTCGBMI",
               "HWTCGISW", "PACCDPAI", "SMKCDSTY", "FVCCDTOT",
               "MEDC_1A", "CCCC_071", "CCCC_91B", "CCCC_101",
               "EDUCDR04", "WTSC_M")
cc21 <- c2[var.names2]
dim(cc21)
```

```r
#> [1] 134072     18
table(cc21$CCCC_051)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          29293         104530              0           208             11
#>     NOT STATED
#>             30
var.names3 <- c("CCCE_051", "CCCE_05A", "CCCE_121", "DHHEGAGE",
               "DHHE_SEX", "INCEGHH", "SDCEGCGT", "HWTEGBMI",
               "HWTEGISW", "PACEDPAI", "SMKEDSTY", "FVCEDTOT",
               "MEDE_1A", "CCCE_071", "CCCE_91F", "CCCE_101",
               "EDUEDR04", "WTSE_M")
cc31 <- c3[var.names3]
dim(cc31)
#> [1] 132221     18
table(cc31$CCCE_051)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          28221         103781              0           191               4
#>     NOT STATED
#>             24
```

Combine 3 cycle datasets

We want to combine data from three different cycles in order to get more subjects in our data. For that, we will have to stack/append data from three different cycles. In order to do so, we need to make the names exarctly the same. E.g., for BMI category, we will rename all 3 variables: `HWTAGSW` (from cycle 1), `HWTCGISW` (from cycle 2) and `HWTEGISW` (from cycle 3) to `bmicat`.

0.0.0.0.1 * Making variable names the same

```r
new.var.names <- c("arthritis", "arthritis.kind", "CVD", "age",
               "sex", "income", "race", "bmi",
               "bmicat", "phyact", "smoke", "fruit",
               "painmed", "ht", "copd", "diab",
               "edu", "weight")
```

```r
names(cc11) <- names(cc21) <- names(cc31) <- new.var.names

table(cc11$arthritis)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          24511         106231              0            110              3
#>     NOT STATED
#>             25
table(cc21$arthritis)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          29293         104530              0            208             11
#>     NOT STATED
#>             30
table(cc31$arthritis)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          28221         103781              0            191              4
#>     NOT STATED
#>             24
```

0.0.0.0.2 * Notice the difference in categorization

Note that, not only the names of the variables are different, sometimes, the categorization labels are also different. Note the BMI (Category) in the three cycles: `HWTAGSW` from cycle 1:

Variable Name	HWTAGSW	**Length**	1	**Position**	70
Question Name					
Concept	Standard weight - (D, G)				
Question					
Universe	Respondents aged 20 to 64 who answered MAMA_037 <> 1				
Note	Based on HWTAGBMI. See documentation on derived variables.				

Content	Code	Sample	Population
UNDERWEIGHT	1	6,040	1,473,150
ACCEPTABLE WEIGHT	2	35,404	7,944,017
OVERWEIGHT	3	44,730	8,882,610
NOT APPLICABLE	6	42,866	7,189,595
NOT STATED	9	1,840	297,962
Total		130,880	25,787,334

`HWTCGISW` from cycle 2:

Variable Name	HWTCGISW	**Length**	1	**Position**	84
Question Name					
Concept	Standard weight - International standard - (D, G)				
Question					
Universe	Respondents aged 20 and over				
Note	Based on HWTCGBMI. See documentation on derived variables.				

Content	Code	Sample	Population
UNDERWEIGHT	1	2,384	550,746
NORMAL WEIGHT	2	49,334	10,539,990
OVERWEIGHT	3	39,972	7,846,692
OBESE	4	19,778	3,522,144
NOT APPLICABLE	6	18,524	3,312,332
NOT STATED	9	4,080	783,526
	Total	134,072	26,555,430

`HWTEGISW` from cycle 3:

Variable Name	HWTEGISW	**Length**	1	**Position**	96
Question Name					
Concept	BMI class. (18 +) / self-report - Intern. standard - (D, G)				
Question					
Universe	Respondents aged 18 and over excluding pregnant women (MAME_037 = 1)				
Note	Based on DHHE_AGE, HWTEDBMI. See documentation on derived variables.				

Content	Code	Sample	Population
UNDERWEIGHT	1	2,918	657,935
NORMAL WEIGHT	2	51,970	11,241,940
OVERWEIGHT	3	40,082	8,126,639
OBESE	4	20,817	3,763,766
NOT APPLICABLE	6	12,317	2,539,752
NOT STATED	9	4,117	796,133
	Total	132,221	27,126,165

```r
table(cc11$bmicat)
#>
#>    UNDERWEIGHT ACCEPT. WEIGHT     OVERWEIGHT NOT APPLICABLE     NOT STATED
#>           6040          35404          44730          42866           1840
table(cc21$bmicat)
#>
#>    UNDERWEIGHT  NORMAL WEIGHT     OVERWEIGHT          OBESE NOT APPLICABLE
#>           2384          49334          39972          19778          18524
#>     DON'T KNOW        REFUSAL     NOT STATED
#>              0              0           4080
table(cc31$bmicat)
#>
#>    UNDERWEIGHT  NORMAL WEIGHT     OVERWEIGHT          OBESE NOT APPLICABLE
#>           2918          51970          40082          20817          12317
#>     DON'T KNOW        REFUSAL     NOT STATED
#>              0              0           4117
```

In cycle 1.1, the second response (code 2) label was named as "Acceptable weight" where as in cycles 2.1 and 3.1 it was named as "Normal weight". Also "obese" category was not present in cycle 1.1.

Similarly look for age categories:

Variable Name	DHHAGAGE	**Length**	2	**Position**	16 - 17
Question Name					
Concept	Age - (G)				
Question					
Universe	All respondents				
Note	Based on DHHA_AGE.				

Content	Code	Sample	Population
12 TO 14 YEARS	1	6,476	1,186,119
15 TO 19 YEARS	2	11,081	2,131,999
20 TO 24 YEARS	3	7,584	2,112,568
25 TO 29 YEARS	4	8,742	2,006,021
30 TO 34 YEARS	5	10,281	2,158,989
35 TO 39 YEARS	6	12,447	2,587,642
40 TO 44 YEARS	7	12,886	2,707,970
45 TO 49 YEARS	8	11,388	2,369,433
50 TO 54 YEARS	9	10,255	2,051,946
55 TO 59 YEARS	10	8,355	1,585,225
60 TO 64 YEARS	11	7,152	1,244,611
65 TO 69 YEARS	12	6,842	1,151,556
70 TO 74 YEARS	13	6,360	1,003,709
75 TO 79 YEARS	14	5,237	740,459
80 YEARS OR OLDER	15	5,794	749,088
	Total	130,880	25,787,334

Variable Name	DHHCGAGE	**Length**	2	**Position**	20 - 21
Question Name					
Concept	Age - (G)				
Question					
Universe	All respondents				
Note					

Content	Code	Sample	Population
12 TO 14 YEARS	1	7,410	1,273,448
15 TO 19 YEARS	2	11,114	2,038,884
20 TO 24 YEARS	3	6,687	2,164,800
25 TO 29 YEARS	4	8,672	2,066,726
30 TO 34 YEARS	5	10,155	2,079,261
35 TO 39 YEARS	6	10,617	2,493,934
40 TO 44 YEARS	7	10,378	2,798,090
45 TO 49 YEARS	8	10,144	2,370,897
50 TO 54 YEARS	9	10,936	2,196,957
55 TO 59 YEARS	10	10,548	1,859,314
60 TO 64 YEARS	11	8,845	1,429,344
65 TO 69 YEARS	12	8,043	1,154,063
70 TO 74 YEARS	13	7,479	1,027,278
75 TO 79 YEARS	14	6,143	804,061
80 YEARS OR OLDER	15	6,901	798,373
	Total	134,072	26,555,430

Variable Name	DHHEGAGE	**Length**	2	**Position**	26 - 27
Question Name	ANC_AGE				
Concept	Age - (G)				
Question	What is your age?				
Universe	All respondents				
Note	Derived from DHHE_DOB, DHHE_MOB and DHHE_YOB during interview and confirmed with respondent.				

Content	Code	Sample	Population
12 TO 14 YEARS	1	6,172	1,235,378
15 TO 17 YEARS	2	6,145	1,304,374
18 TO 19 YEARS	3	3,989	813,009
20 TO 24 YEARS	4	7,740	2,239,016
25 TO 29 YEARS	5	9,227	2,103,064
30 TO 34 YEARS	6	10,252	2,079,630
35 TO 39 YEARS	7	10,058	2,282,458
40 TO 44 YEARS	8	11,172	2,803,793
45 TO 49 YEARS	9	9,143	2,532,177
50 TO 54 YEARS	10	10,296	2,251,247
55 TO 59 YEARS	11	10,645	1,973,801
60 TO 64 YEARS	12	9,268	1,580,369
65 TO 69 YEARS	13	7,846	1,213,621
70 TO 74 YEARS	14	7,124	1,030,975
75 TO 79 YEARS	15	5,961	807,900
80 YEARS OR OLDER	16	7,183	875,354
	Total	132,221	27,126,165

Similarly look for income categories:

Variable Name	INCAGHH	**Length**	2	**Position**	824 - 825
Question Name					
Concept	Total household income from all sources - (D, G)				
Question					
Universe	All respondents				
Note	Based on INCA_3A to INCA_3G. See documentation on derived variables.				

Content	Code	Sample	Population
NO INCOME	1	506	116,955
LESS THAN $15,000	2	14,103	1,865,128
$15,000 TO $29,999	3	22,689	3,692,069
$30,000 TO $49,999	4	27,169	5,201,039
$50,000 TO $79,999	5	29,281	6,327,658
$80,000 OR MORE	6	22,647	5,785,988
NOT STATED	99	14,485	2,798,497
	Total	130,880	25,787,334

Variable Name	INCCGHH	**Length**	1	**Position**	1366
Question Name					
Concept	Total household income from all sources - (D, G)				
Question					
Universe	All respondents				
Note	Based on INCC_3A, INCC_3B, INCC_3C, INCC_3D, INCC_3E, INCC_3F, INCC_3G. See documentation on derived variables.				

Content	Code	Sample	Population
NO INCOME AND LESS THAN $15,000	1	12,130	1,492,803
$15,000 TO $29,999	2	20,710	3,068,212
$30,000 TO $49,999	3	25,593	4,679,063
$50,000 TO $79,999	4	27,779	6,004,218
$80,000 OR MORE	5	25,129	6,763,874
NOT STATED	9	22,731	4,547,260
	Total	134,072	26,555,430

Variable Name	INCEGHH	**Length**	1	**Position**	1587
Question Name					
Concept	Total household income from all sources - (D, G)				
Question					
Universe	All respondents				
Note	Based on INCE_3A, INCE_3B, INCE_3C, INCE_3D, INCE_3E, INCE_3F, INCE_3G. See documentation on derived variables.				

Content	Code	Sample	Population
NO INCOME AND LESS THAN $15,000	1	10,826	1,282,889
$15,000 TO $29,999	2	20,367	2,928,006
$30,000 TO $49,999	3	25,178	4,548,210
$50,000 TO $79,999	4	28,048	6,223,197
$80,000 OR MORE	5	27,939	7,865,909
NOT STATED	9	19,863	4,277,953
Total		132,221	27,126,165

Therefore, we need to recode the variable value labels carefully.

0.0.0.0.3 * Appending

Below we append all the three cycles of data:

```
cc123a <- rbind(cc11,cc21,cc31)
```

Subsetting according to eligibility criteria

Criteria 1: control group

Exposure group is people with osteoarthritis. The control group is people who do not have osteoarthritis.

In cycle 1, there were 2 related questions: - Do you have arthritis or rheumatism excluding fibromyalgia? (variable `arthritis`) - What kind of arthritis do you have? (variable `arthritis.kind`)

```
table(cc123a$arthritis)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          82025         314542              0            509             18
#>     NOT STATED
#>             79
table(cc123a$arthritis.kind)
#>
```

```
#> RHEUMATOID ARTH  OSTEOARTHRITIS           OTHER  NOT APPLICABLE      DON'T KNOW
#>           19099           40943            7305          314542           12354
#>         REFUSAL      NOT STATED      RHEUMATISM
#>             215             619            2096
dim(cc123a)
#> [1] 397173     18
```

In the control group, we do not want to put people with other types of arthritis.

```
c123sub1 <- subset(cc123a, arthritis.kind == "OSTEOARTHRITIS" |
                     arthritis.kind == "NOT APPLICABLE" )
dim(c123sub1)
#> [1] 355485     18
table(c123sub1$arthritis.kind)
#>
#> RHEUMATOID ARTH  OSTEOARTHRITIS           OTHER  NOT APPLICABLE      DON'T KNOW
#>               0           40943               0          314542               0
#>         REFUSAL      NOT STATED      RHEUMATISM
#>               0               0               0
table(c123sub1$arthritis)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          40943         314542              0              0              0
#>     NOT STATED
#>              0
require(car)
#> Loading required package: car
#> Loading required package: carData
c123sub1$arthritis.kind <- recode(c123sub1$arthritis.kind,
                        "'OSTEOARTHRITIS'='OA';
                         'NOT APPLICABLE'='Control';
                         else=NA",
                         as.factor = FALSE)
table(c123sub1$arthritis.kind, useNA = "always")
#>
#> Control      OA    <NA>
#>  314542   40943       0
c123sub1$OA <- c123sub1$arthritis.kind
c123sub1$arthritis.kind <- NULL
```

```r
c123sub1$arthritis <- NULL
```

Criteria 2: retain valid responses for outcome

```r
table(c123sub1$CVD)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          19177         336021              0            252             35
#>     NOT STATED
#>              0
c123sub2 <- subset(c123sub1, CVD == "YES" | CVD == "NO")
table(c123sub2$CVD)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          19177         336021              0              0              0
#>     NOT STATED
#>              0
dim(c123sub2)
#> [1] 355198     17
c123sub2$CVD <- recode(c123sub2$CVD,
                       "'YES'='event';
                        'NO'='0 event';
                        else=NA",
                        as.factor = FALSE)
table(c123sub2$CVD, useNA = "always")
#>
#> 0 event   event    <NA>
#>  336021   19177       0
```

Criteria 3: Is there a zero cell?

Check out 'Universe' for all the variables under consideration. Is there a possibility that cross-tabulation of some of the categories will produce zero?

For example, the 'Universe' for BMI or BMI category includes 'Respondents aged 20 to 64'.

Variable Name	HWTAGBMI	**Length**	5.1	**Position**	65 - 69
Question Name					
Concept	Body Mass Index - (D, G)				
Question					
Universe	Respondents aged 20 to 64 who answered MAMA_037 <> 1				
Note	Based on HWTADHTM, HWRADWTK and MAMA_037. See documentation on derived variables.				

Content	Code	Sample	Population
BMI SCORE	14.0 - 57.6	86,174	18,299,777
NOT APPLICABLE	999.6	42,866	7,189,595
NOT STATED	999.9	1,840	297,962
	Total	130,880	25,787,334

Variable Name	HWTAGSW	**Length**	1	**Position**	70
Question Name					
Concept	Standard weight - (D, G)				
Question					
Universe	Respondents aged 20 to 64 who answered MAMA_037 <> 1				
Note	Based on HWTAGBMI. See documentation on derived variables.				

Content	Code	Sample	Population
UNDERWEIGHT	1	6,040	1,473,150
ACCEPTABLE WEIGHT	2	35,404	7,944,017
OVERWEIGHT	3	44,730	8,882,610
NOT APPLICABLE	6	42,866	7,189,595
NOT STATED	9	1,840	297,962
	Total	130,880	25,787,334

Therefore, we will not have BMI from anyone aged less than 20 or over 64.

```
table(c123sub2$bmicat, c123sub2$age)[,1:2]
#>
#>                  12 TO 14 YEARS 15 TO 19 YEARS
#>   UNDERWEIGHT                 0              0
#>   ACCEPT. WEIGHT              0              0
#>   OVERWEIGHT                  0              0
#>   NOT APPLICABLE          19934          21891
#>   NOT STATED                  0              0
#>   NORMAL WEIGHT               0              0
#>   OBESE                       0              0
#>   DON'T KNOW                  0              0
#>   REFUSAL                     0              0
# Note the categories of bmicat (duplicate categories)
```

Also, we check the prevalence of OA and CVD for subjects less than 20 years of age: not a lot of people (still potential for a sensitivity analysis).

```r
table(c123sub2$OA, c123sub2$age)
#>
#>           12 TO 14 YEARS 15 TO 19 YEARS 20 TO 24 YEARS 25 TO 29 YEARS
#>   Control          19927          21795          21303          25490
#>   OA                   7             96            191            345
#>
#>           30 TO 34 YEARS 35 TO 39 YEARS 40 TO 44 YEARS 45 TO 49 YEARS
#>   Control          28851          30359          30267          25302
#>   OA                 595           1027           1639           2482
#>
#>           50 TO 54 YEARS 55 TO 59 YEARS 60 TO 64 YEARS 65 TO 69 YEARS
#>   Control          23670          20036          15751          13024
#>   OA                3988           4991           5093           5257
#>
#>           70 TO 74 YEARS 75 TO 79 YEARS 80 YEARS OR MORE NOT APPLICABLE
#>   Control          11011           8426             9127              0
#>   OA                5236           4671             5227              0
#>
#>           DON'T KNOW REFUSAL NOT STATED 15 TO 17 YEARS 18 TO 19 YEARS
#>   Control          0       0          0           6084           3907
#>   OA               0       0          0              8             15
table(c123sub2$CVD, c123sub2$age)
#>
#>           12 TO 14 YEARS 15 TO 19 YEARS 20 TO 24 YEARS 25 TO 29 YEARS
#>   0 event          19853          21776          21354          25670
#>   event               81            115            140            165
#>
#>           30 TO 34 YEARS 35 TO 39 YEARS 40 TO 44 YEARS 45 TO 49 YEARS
#>   0 event          29237          31090          31378          27049
#>   event              209            296            528            735
#>
#>           50 TO 54 YEARS 55 TO 59 YEARS 60 TO 64 YEARS 65 TO 69 YEARS
#>   0 event          26506          23290          18762          15789
#>   event             1152           1737           2082           2492
#>
#>           70 TO 74 YEARS 75 TO 79 YEARS 80 YEARS OR MORE NOT APPLICABLE
#>   0 event          13375          10229            10703              0
#>   event             2872           2868             3651              0
#>
#>           DON'T KNOW REFUSAL NOT STATED 15 TO 17 YEARS 18 TO 19 YEARS
```

```
#>   0 event          0       0          0           6066            3894
#>   event            0       0          0             26              28
```

Accordingly, we will restrict our analysis (and aim) to adult target population only (age 20 and +). For that, first, recode the age variable, and then exclude the `teen` category.

```
# CCHS cycle 1.1 has: '15 TO 19 YEARS'
# Other cycles have: '15 TO 17 YEARS', '18 TO 19 YEARS'
c123sub2$age <- recode(c123sub2$age,
                        "c('12 TO 14 YEARS','15 TO 19 YEARS',
                         '15 TO 17 YEARS', '18 TO 19 YEARS')='teen';
                         c('20 TO 24 YEARS','25 TO 29 YEARS',
                           '30 TO 34 YEARS','35 TO 39 YEARS')='20-39 years';
                         c('40 TO 44 YEARS','45 TO 49 YEARS')='40-49 years';
                         c('50 TO 54 YEARS','55 TO 59 YEARS')='50-59 years';
                         c('60 TO 64 YEARS')='60-64 years';
                         else='65 years and over'",
                         as.factor = FALSE)
table(c123sub2$age)
#>
#>       20-39 years       40-49 years       50-59 years       60-64 years
#>            108161             59690             52685             20844
#> 65 years and over              teen
#>             61979             51839
dim(c123sub2)
#> [1] 355198     17
c123sub3 <- subset(c123sub2, age != 'teen' &
                     age != '65 years and over')
table(c123sub3$age, useNA = "always")
#>
#> 20-39 years 40-49 years 50-59 years 60-64 years        <NA>
#>      108161       59690       52685       20844           0
dim(c123sub3)
#> [1] 241380     17
```

Criteria 4: Assign missing to the invalid covariate responses

We have invalid covariate responses, such as refused, not ascertained, and don't know responses. Let's assign missing values

for these invalid responses.

```
# sex
table(c123sub3$sex)
#>
#>           MALE         FEMALE NOT APPLICABLE     NOT STATED     DON'T KNOW
#>         114104         127276              0              0              0
#>        REFUSAL
#>              0
c123sub3$sex <- car::recode(c123sub3$sex,
                        "'MALE'='Male';
                         'FEMALE' = 'Female';
                         else = NA",
                         as.factor = FALSE)
table(c123sub3$sex, useNA = "always")
#>
#> Female   Male   <NA>
#> 127276 114104      0
# Race
table(c123sub3$race)
#>
#>            WHITE VISIBLE MINORITY   NOT APPLICABLE       NOT STATED
#>           210307            25840                0             5233
#>       DON'T KNOW          REFUSAL
#>                0                0
c123sub3$race <- car::recode(c123sub3$race,
                        "'WHITE'='White';
                         'VISIBLE MINORITY' = 'Non-white';
                         else = NA",
                         as.factor = FALSE)
table(c123sub3$race, useNA = "always")
#>
#> Non-white     White      <NA>
#>     25840    210307      5233

# income
table(c123sub3$income)
#>
#>        NO INCOME LESS THAN 15,000  $15,000-$29,999  $30,000-$49,999
#>             5953             6985            29888            49496
#>  $50,000-$79,999  $80,000 OR MORE   NOT APPLICABLE       NOT STATED
```

```
#>            61093            57056                0            25730
#>       DON'T KNOW          REFUSAL   NO OR <$15,000
#>                0                0             5179
# cycle 1.1 has: 'NO INCOME','LESS THAN 15,000'
# Other cycles have: 'NO OR <$15,000'
c123sub3$income <- car::recode(c123sub3$income,
                        "c('NO OR <$15,000', 'NO INCOME',
                        'LESS THAN 15,000',
                        '$15,000-$29,999')='$29,999 or less';
                        '$30,000-$49,999' = '$30,000-$49,999';
                        '$50,000-$79,999' = '$50,000-$79,999';
                        '$80,000 OR MORE' = '$80,000 or more';
                        else = NA",
                          as.factor = FALSE)
table(c123sub3$income, useNA = "always")
#>
#> $29,999 or less $30,000-$49,999 $50,000-$79,999 $80,000 or more            <NA>
#>           48005           49496           61093           57056           25730
# BMI
table(c123sub3$bmicat)
#>
#>    UNDERWEIGHT ACCEPT. WEIGHT     OVERWEIGHT NOT APPLICABLE     NOT STATED
#>           8964          33100          93107           1038           7577
#>  NORMAL WEIGHT          OBESE     DON'T KNOW        REFUSAL
#>          70278          27316              0              0
c123sub3$bmicat <- car::recode(c123sub3$bmicat,
                        "'UNDERWEIGHT'='Underweight';
                         c('ACCEPT. WEIGHT','NORMAL WEIGHT')='Normal';
                         c('OVERWEIGHT','OBESE') = 'Overweight';
                         else = NA",
                         as.factor = FALSE)
table(c123sub3$bmicat, useNA = "always")
#>
#>      Normal  Overweight Underweight        <NA>
#>      103378      120423        8964        8615
c123sub3$bmi <- NULL # no need of the original BMI values
# physical activity
table(c123sub3$phyact)
#>
#>         ACTIVE       MODERATE       INACTIVE NOT APPLICABLE     NOT STATED
```

```
#>          57033          60164         117516              0            6667
#>     DON'T KNOW        REFUSAL
#>              0              0
c123sub3$phyact <- car::recode(c123sub3$phyact,
                        "'ACTIVE'='Active';
                         'MODERATE' = 'Moderate';
                         'INACTIVE' = 'Inactive';
                         else = NA",
                         as.factor = FALSE)
table(c123sub3$phyact, useNA = "always")
#>
#>   Active Inactive Moderate     <NA>
#>    57033   117516    60164     6667
# smoking
table(c123sub3$smoke)
#>
#>            DAILY       OCCASIONAL ALWAYS OCCASION.     FORMER DAILY
#>            58747             8175             4399            59722
#> FORMER OCCASION.     NEVER SMOKED   NOT APPLICABLE       NOT STATED
#>            38123            71397                0              817
#>       DON'T KNOW          REFUSAL
#>                0                0
c123sub3$smoke <- car::recode(c123sub3$smoke,
                        "c('DAILY','OCCASIONAL',
                           'ALWAYS OCCASION.')='Current smoker';
                         c('FORMER DAILY','FORMER OCCASION.',
                           'ALWAYS OCCASION.') = 'Former smoker';
                         'NEVER SMOKED' = 'Never smoker';
                         else = NA",
                         as.factor = FALSE)
table(c123sub3$smoke, useNA = "always")
#>
#> Current smoker  Former smoker   Never smoker           <NA>
#>          71321          97845          71397            817
# fruit and vegetable consumption
str(c123sub3$fruit)
#>  Factor w/ 303 levels "0","0.1","0.2",..: 42 67 39 61 51 51 48 41 17 27 ...
c123sub3$fruit.cont <- c123sub3$fruit
c123sub3$fruit2 <- as.numeric(as.character(c123sub3$fruit))
#> Warning: NAs introduced by coercion
```

```r
# Note: do not use as.numeric(c123sub3$fruit)
summary(c123sub3$fruit2)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>    0.00    2.90    4.10    4.63    5.90   64.30   42987
c123sub3$fruit2 <- cut(c123sub3$fruit2,
                       breaks = c(0,3,6,Inf),
                       right = TRUE,
                       labels = c("0-3 daily serving",
                                  "4-6 daily serving",
                                  "6+ daily serving"))
table(c123sub3$fruit2, useNA = "always")
#>
#> 0-3 daily serving 4-6 daily serving  6+ daily serving              <NA>
#>             56256             96177             45861             43086
c123sub3$fruit <- c123sub3$fruit2
c123sub3$fruit2 <- NULL
# pain medication
table(c123sub3$painmed)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          25743          11141         204323             32             13
#>     NOT STATED
#>            128
c123sub3$painmed <- car::recode(c123sub3$painmed,
                        "'YES'='Yes';
                         'NO' = 'No';
                         else = NA",
                         as.factor = FALSE)
table(c123sub3$painmed, useNA = "always")
#>
#>     No    Yes   <NA>
#>  11141  25743 204496
# hypertension
table(c123sub3$ht)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          27592         213432              0            331             25
#>     NOT STATED
#>              0
c123sub3$ht <- car::recode(c123sub3$ht,
```

```
                        "'YES'='Yes';
                         'NO' = 'No';
                         else = NA",
                         as.factor = FALSE)
table(c123sub3$ht, useNA = "always")
#>
#>     No    Yes   <NA>
#> 213432  27592    356
# COPD
table(c123sub3$copd)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>           1353         192608          47329             70              0
#>     NOT STATED
#>             20
c123sub3$copd <- car::recode(c123sub3$copd,
                        "'YES'='Yes';
                         'NO' = 'No';
                         else = NA",
                         as.factor = FALSE)
table(c123sub3$copd, useNA = "always")
#>
#>     No    Yes   <NA>
#> 192608   1353  47419
# Diabetes
table(c123sub3$diab)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>           8811         232486              0             81              2
#>     NOT STATED
#>              0
c123sub3$diab <- car::recode(c123sub3$diab,
                        "'YES'='Yes';
                         'NO' = 'No';
                         else = NA",
                         as.factor = FALSE)
table(c123sub3$diab, useNA = "always")
#>
#>     No    Yes   <NA>
#> 232486   8811     83
```

```r
# Education
table(c123sub3$edu)
#>
#> < THAN SECONDARY  SECONDARY GRAD.  OTHER POST-SEC.  POST-SEC. GRAD.
#>            37775            44376            19273           136031
#>   NOT APPLICABLE       NOT STATED       DON'T KNOW          REFUSAL
#>                0             3925                0                0
c123sub3$edu <- car::recode(c123sub3$edu,
                        "'< THAN SECONDARY'='< 2ndary';
                         'SECONDARY GRAD.' = '2nd grad.';
                         'POST-SEC. GRAD.' = 'Post-2nd grad.';
                         'OTHER POST-SEC.' = 'Other 2nd grad.';
                         else = NA",
                         as.factor = FALSE)
table(c123sub3$edu, useNA = "always")
#>
#>        < 2ndary       2nd grad. Other 2nd grad.  Post-2nd grad.            <NA>
#>           37775           44376           19273         136031            3925
```

Naive Analysis of combined 3 cycles

In the current analysis, we will simply consider all of the variables under consideration as 'confounders', and include in our analysis. Later we will perform a refined analysis.

Summary of the analytic data

Including missing values

```r
dim(c123sub3)
#> [1] 241380     17
analytic <- c123sub3
dim(analytic)
#> [1] 241380     17

require("tableone")
#> Loading required package: tableone
```

```r
CreateTableOne(vars = c("CVD", "age",
               "sex", "income", "race",
               "bmicat", "phyact", "smoke", "fruit",
               "painmed", "ht", "copd", "diab", "edu"),
               data = analytic, includeNA = TRUE)
#>
#>                        Overall
#>   n                    241380
#>   CVD = event (%)        7044 ( 2.9)
#>   age (%)
#>      20-39 years       108161 (44.8)
#>      40-49 years        59690 (24.7)
#>      50-59 years        52685 (21.8)
#>      60-64 years        20844 ( 8.6)
#>   sex = Male (%)       114104 (47.3)
#>   income (%)
#>      $29,999 or less    48005 (19.9)
#>      $30,000-$49,999    49496 (20.5)
#>      $50,000-$79,999    61093 (25.3)
#>      $80,000 or more    57056 (23.6)
#>      NA                 25730 (10.7)
#>   race (%)
#>      Non-white          25840 (10.7)
#>      White             210307 (87.1)
#>      NA                  5233 ( 2.2)
#>   bmicat (%)
#>      Normal            103378 (42.8)
#>      Overweight        120423 (49.9)
#>      Underweight         8964 ( 3.7)
#>      NA                  8615 ( 3.6)
#>   phyact (%)
#>      Active             57033 (23.6)
#>      Inactive          117516 (48.7)
#>      Moderate           60164 (24.9)
#>      NA                  6667 ( 2.8)
#>   smoke (%)
#>      Current smoker     71321 (29.5)
#>      Former smoker      97845 (40.5)
#>      Never smoker       71397 (29.6)
#>      NA                   817 ( 0.3)
```

```
#>   fruit (%)
#>      0-3 daily serving  56256 (23.3)
#>      4-6 daily serving  96177 (39.8)
#>      6+ daily serving   45861 (19.0)
#>      NA                 43086 (17.8)
#>   painmed (%)
#>      No                 11141 ( 4.6)
#>      Yes                25743 (10.7)
#>      NA                204496 (84.7)
#>   ht (%)
#>      No                213432 (88.4)
#>      Yes                27592 (11.4)
#>      NA                   356 ( 0.1)
#>   copd (%)
#>      No                192608 (79.8)
#>      Yes                 1353 ( 0.6)
#>      NA                 47419 (19.6)
#>   diab (%)
#>      No                232486 (96.3)
#>      Yes                 8811 ( 3.7)
#>      NA                    83 ( 0.0)
#>   edu (%)
#>      < 2ndary           37775 (15.6)
#>      2nd grad.          44376 (18.4)
#>      Other 2nd grad.    19273 ( 8.0)
#>      Post-2nd grad.    136031 (56.4)
#>      NA                  3925 ( 1.6)
CreateTableOne(vars = c("CVD", "age",
               "sex", "income", "race",
               "bmicat", "phyact", "smoke", "fruit",
               "painmed", "ht", "copd", "diab", "edu"),
               data = analytic, strata = "OA", includeNA = TRUE)
#>                         Stratified by OA
#>                          Control        OA            p      test
#>   n                      221029         20351
#>   CVD = event (%)          5429 ( 2.5)   1615 ( 7.9)  <0.001
#>   age (%)                                             <0.001
#>      20-39 years        106003 (48.0)   2158 (10.6)
#>      40-49 years         55569 (25.1)   4121 (20.2)
#>      50-59 years         43706 (19.8)   8979 (44.1)
```

```
#>      60-64 years        15751 ( 7.1)   5093 (25.0)
#>   sex = Male (%)       107729 (48.7)   6375 (31.3)  <0.001
#>   income (%)                                        <0.001
#>      $29,999 or less    42019 (19.0)   5986 (29.4)
#>      $30,000-$49,999    45090 (20.4)   4406 (21.7)
#>      $50,000-$79,999    56754 (25.7)   4339 (21.3)
#>      $80,000 or more    53637 (24.3)   3419 (16.8)
#>      NA                 23529 (10.6)   2201 (10.8)
#>   race (%)                                          <0.001
#>      Non-white          24681 (11.2)   1159 ( 5.7)
#>      White             191513 (86.6)  18794 (92.3)
#>      NA                  4835 ( 2.2)    398 ( 2.0)
#>   bmicat (%)                                        <0.001
#>      Normal             96697 (43.7)   6681 (32.8)
#>      Overweight        107871 (48.8)  12552 (61.7)
#>      Underweight         8490 ( 3.8)    474 ( 2.3)
#>      NA                  7971 ( 3.6)    644 ( 3.2)
#>   phyact (%)                                        <0.001
#>      Active             52942 (24.0)   4091 (20.1)
#>      Inactive          106580 (48.2)  10936 (53.7)
#>      Moderate           55222 (25.0)   4942 (24.3)
#>      NA                  6285 ( 2.8)    382 ( 1.9)
#>   smoke (%)                                         <0.001
#>      Current smoker     65398 (29.6)   5923 (29.1)
#>      Former smoker      88210 (39.9)   9635 (47.3)
#>      Never smoker       66663 (30.2)   4734 (23.3)
#>      NA                   758 ( 0.3)     59 ( 0.3)
#>   fruit (%)                                         <0.001
#>      0-3 daily serving  52140 (23.6)   4116 (20.2)
#>      4-6 daily serving  87951 (39.8)   8226 (40.4)
#>      6+ daily serving   41606 (18.8)   4255 (20.9)
#>      NA                 39332 (17.8)   3754 (18.4)
#>   painmed (%)                                       <0.001
#>      No                 10624 ( 4.8)    517 ( 2.5)
#>      Yes                23084 (10.4)   2659 (13.1)
#>      NA                187321 (84.7)  17175 (84.4)
#>   ht (%)                                            <0.001
#>      No                198550 (89.8)  14882 (73.1)
#>      Yes                22142 (10.0)   5450 (26.8)
#>      NA                   337 ( 0.2)     19 ( 0.1)
```

```
#>   copd (%)                                                  <0.001
#>      No                 173224 (78.4)  19384 (95.2)
#>      Yes                   938 ( 0.4)    415 ( 2.0)
#>      NA                  46867 (21.2)    552 ( 2.7)
#>   diab (%)                                                  <0.001
#>      No                 213910 (96.8)  18576 (91.3)
#>      Yes                  7046 ( 3.2)   1765 ( 8.7)
#>      NA                     73 ( 0.0)     10 ( 0.0)
#>   edu (%)                                                   <0.001
#>      < 2ndary            32884 (14.9)   4891 (24.0)
#>      2nd grad.           40950 (18.5)   3426 (16.8)
#>      Other 2nd grad.     17808 ( 8.1)   1465 ( 7.2)
#>      Post-2nd grad.     125772 (56.9)  10259 (50.4)
#>      NA                   3615 ( 1.6)    310 ( 1.5)
require(DataExplorer)
#> Loading required package: DataExplorer
plot_missing(analytic)
```

Let us investigate why pain medication has so much missing

Variable Name	DRGA_1A	**Length**	1	**Position**	193
Question Name	DG_Q1A				
Concept	Medication - pain relievers				
Question	In the past month, did you take pain relievers such as aspirin or Tylenol (including arthritis medicine and anti-inflammatories)?				
Universe	All respondents with DRGAFOPT = 1				
Note					

Content	Code	Sample	Population
YES	1	21,775	5,601,997
NO	2	9,358	2,682,979
NOT APPLICABLE	6	99,693	17,489,866
DON'T KNOW	7	24	7,030
REFUSAL	8	7	1,949
NOT STATED	9	23	3,514
	Total	130,880	25,787,334

Variable Name	MEDC_1A	**Length**	1	**Position**	1169
Question Name	MED_Q1A				
Concept	Medication - pain relievers - past mo.				
Question	In the past month, did you take pain relievers such as aspirin or Tylenol (including arthritis medicine and anti-inflammatories)?				
Universe	Respondents with MEDCFOPT = 1				
Note	Formerly DRGA_1A in cycle 1.1, Optional Content - see Survey Content section in User Guide.				

Content	Code	Sample	Population
YES	1	15,061	2,838,591
NO	2	6,430	1,266,989
NOT APPLICABLE	6	112,317	22,397,702
DON'T KNOW	7	31	7,403
REFUSAL	8	31	7,213
NOT STATED	9	202	37,532
	Total	134,072	26,555,430

Variable Name	MEDE_1A	**Length**	1	**Position**	188
Question Name	MED_Q1A				
Concept	Medication - pain relievers - past mo.				
Question	(In the past month, did you take) pain relievers such as aspirin or Tylenol (including arthritis medicine and anti-inflammatories)?				
Universe	Respondents with MEDEFOPT = 1				
Note	Formerly DRGA_1A in cycle 1.1.				

Content	Code	Sample	Population
YES	1	6,059	829,908
NO	2	3,111	413,254
NOT APPLICABLE	6	123,044	25,881,369
DON'T KNOW	7	7	1,633
	Total	132,221	27,126,165

Optional content respondent (cycle 3.1):

Variable Name	MEDEFOPT	**Length**	1	**Position**	187
Question Name					
Concept	Optional module: Medication use - (F)				
Question					
Universe	All respondents				
Note					

Content	Code	Sample	Population
YES	1	9,177	1,244,796
NO	2	123,044	25,881,369
	Total	132,221	27,126,165

In cycle 2.1, only 21,755 out of 134,072 responded to optional medication component.

Complete case analysis

```
dim(c123sub3)
#> [1] 241380     17
analytic2 <- as.data.frame(na.omit(c123sub3))
dim(analytic2)
#> [1] 21623    17


tab1 <- CreateTableOne(vars = c("CVD", "age",
               "sex", "income", "race",
               "bmicat", "phyact", "smoke", "fruit",
               "painmed", "ht", "copd", "diab", "edu"),
               data = analytic2, includeNA = TRUE)
print(tab1, showAllLevels = TRUE)
#>
#>                level             Overall
#>   n                              21623
#>   CVD (%)      0 event           20917 (96.7)
#>                event               706 ( 3.3)
#>   age (%)      20-39 years        7119 (32.9)
#>                40-49 years        7024 (32.5)
#>                50-59 years        5457 (25.2)
#>                60-64 years        2023 ( 9.4)
#>   sex (%)      Female            10982 (50.8)
#>                Male              10641 (49.2)
#>   income (%)   $29,999 or less    4054 (18.7)
#>                $30,000-$49,999    4461 (20.6)
#>                $50,000-$79,999    6600 (30.5)
#>                $80,000 or more    6508 (30.1)
#>   race (%)     Non-white          2488 (11.5)
#>                White             19135 (88.5)
#>   bmicat (%)   Normal             8993 (41.6)
#>                Overweight        11739 (54.3)
#>                Underweight         891 ( 4.1)
#>   phyact (%)   Active             5502 (25.4)
```

```
#>                Inactive          10495 (48.5)
#>                Moderate           5626 (26.0)
#>   smoke (%)    Current smoker     5887 (27.2)
#>                Former smoker      9368 (43.3)
#>                Never smoker       6368 (29.5)
#>   fruit (%)    0-3 daily serving  5806 (26.9)
#>                4-6 daily serving 10730 (49.6)
#>                6+ daily serving   5087 (23.5)
#>   painmed (%) No                  6197 (28.7)
#>                Yes               15426 (71.3)
#>   ht (%)       No                19014 (87.9)
#>                Yes                2609 (12.1)
#>   copd (%)     No                21475 (99.3)
#>                Yes                 148 ( 0.7)
#>   diab (%)     No                20760 (96.0)
#>                Yes                 863 ( 4.0)
#>   edu (%)      < 2ndary           2998 (13.9)
#>                2nd grad.          4605 (21.3)
#>                Other 2nd grad.    1509 ( 7.0)
#>                Post-2nd grad.    12511 (57.9)
tab1b <- CreateTableOne(vars = c("CVD", "age",
               "sex", "income", "race",
               "bmicat", "phyact", "smoke", "fruit",
               "painmed", "ht", "copd", "diab", "edu"),
               data = analytic2, strata = "OA", includeNA = TRUE)
print(tab1b, showAllLevels = TRUE)
#>               Stratified by OA
#>                level             Control       OA           p      test
#>   n                             19459         2164
#>   CVD (%)      0 event           18917 (97.2)  2000 (92.4)  <0.001
#>                event               542 ( 2.8)   164 ( 7.6)
#>   age (%)      20-39 years        6915 (35.5)   204 ( 9.4)  <0.001
#>                40-49 years        6515 (33.5)   509 (23.5)
#>                50-59 years        4504 (23.1)   953 (44.0)
#>                60-64 years        1525 ( 7.8)   498 (23.0)
#>   sex (%)      Female             9521 (48.9)  1461 (67.5)  <0.001
#>                Male               9938 (51.1)   703 (32.5)
#>   income (%)   $29,999 or less    3413 (17.5)   641 (29.6)  <0.001
#>                $30,000-$49,999    3968 (20.4)   493 (22.8)
#>                $50,000-$79,999    6023 (31.0)   577 (26.7)
```

```
#>                 $80,000 or more    6055 (31.1)   453 (20.9)
#>   race (%)      Non-white          2370 (12.2)   118 ( 5.5)  <0.001
#>                 White             17089 (87.8)  2046 (94.5)
#>   bmicat (%)    Normal             8277 (42.5)   716 (33.1)  <0.001
#>                 Overweight        10356 (53.2)  1383 (63.9)
#>                 Underweight         826 ( 4.2)    65 ( 3.0)
#>   phyact (%)    Active             4986 (25.6)   516 (23.8)   0.190
#>                 Inactive           9417 (48.4)  1078 (49.8)
#>                 Moderate           5056 (26.0)   570 (26.3)
#>   smoke (%)     Current smoker     5247 (27.0)   640 (29.6)  <0.001
#>                 Former smoker      8363 (43.0)  1005 (46.4)
#>                 Never smoker       5849 (30.1)   519 (24.0)
#>   fruit (%)     0-3 daily serving  5290 (27.2)   516 (23.8)  <0.001
#>                 4-6 daily serving  9686 (49.8)  1044 (48.2)
#>                 6+ daily serving   4483 (23.0)   604 (27.9)
#>   painmed (%)   No                 5859 (30.1)   338 (15.6)  <0.001
#>                 Yes               13600 (69.9)  1826 (84.4)
#>   ht (%)        No                17356 (89.2)  1658 (76.6)  <0.001
#>                 Yes                2103 (10.8)   506 (23.4)
#>   copd (%)      No                19359 (99.5)  2116 (97.8)  <0.001
#>                 Yes                 100 ( 0.5)    48 ( 2.2)
#>   diab (%)      No                18751 (96.4)  2009 (92.8)  <0.001
#>                 Yes                 708 ( 3.6)   155 ( 7.2)
#>   edu (%)       < 2ndary           2527 (13.0)   471 (21.8)  <0.001
#>                 2nd grad.          4173 (21.4)   432 (20.0)
#>                 Other 2nd grad.    1364 ( 7.0)   145 ( 6.7)
#>                 Post-2nd grad.    11395 (58.6)  1116 (51.6)
```

Save data for later

```
save(analytic, analytic2, cc123a, file = "Data/researchquestion/OA123CVD.RData")
```

References

Causal question-2

Working with a causal question using NHANES

We are interested in exploring the relationship between diabetes (binary exposure variable defined as whether the doctor ever told the participant has diabetes) and cholesterol (binary outcome variable defined as whether total cholesterol is more than 200 mg/dL). Below is the PICOT:

PICOT element	Description
P	US adults
I	Diabetes
C	No diabetes
O	Total cholesterol > 200 mg/dL
T	2017–2018

First, we will prepare the analytic dataset from NHANES 2017–2018.

Second, we will work with subset of data to assess the association between diabetes and cholesterol, and to get proper SE and 95% CI for the estimate. We emphasize the correct usage of the survey's design features (correct handling of survey design elements, such as stratification, clustering, and weighting) to obtain accurate population-level estimates.

```r
# Load required packages
require(SASxport)
require(DiagrammeR)
require(DiagrammeRsvg)
require(rsvg)
library(magrittr)
library(svglite)
```

```r
library(png)
require(nhanesA)
require(survey)
require(Publish)
require(jtools)
```

Steps for creating analytic dataset

We will combine multiple components (e.g., demographic, blood pressure) using the unique identifier to create our analytic dataset.

Within NHANES datasets in a given cycle, each sampled person has an unique identifier sequence number (variable SEQN).

Download and Subsetting to retain only the useful variables

Search literature for the relevant variables, and then see if some of them are available in the NHANES data.

An an example, let us assume that variables listed in the following figures are known to be useful. Then we will try to indentify, in which NHANES component we have these variables.

Peters, Fabian, and Levy (2014)

Refer to the earlier chapter to get a more detailed understanding of how we search for variables within NHANES.

education
marital status
weights/psu/strata
gender
Demographics
age
bornCountry
race
systolic
Blood pressure
diastolic
BMIcat
weist
bodyweight
Body Measures
heignt
BMI
Smoking
smoking
ID
Analytic Data
Cholesterol
cholesterol
bilirubin
phosphorus
sodium
potassium
Standard Biochemistry Profile
globulin
calcium
triglycerides
uric acid
protein
Physical Activity
Recreational activity
Work acticity
Diabetes
Have diabetes

Demographic component:

```r
demo <- nhanes('DEMO_J') # Both males and females 0 YEARS - 150 YEARS
demo <- demo[c("SEQN", # Respondent sequence number
                "RIAGENDR", # gender
                "RIDAGEYR", # Age in years at screening
                "DMDBORN4", # Country of birth
                "RIDRETH3", # Race/Hispanic origin w/ NH Asian
                "DMDEDUC3", # Education level - Children/Youth 6-19
                "DMDEDUC2", # Education level - Adults 20+
                "DMDMARTL", # Marital status: 20 YEARS - 150 YEARS
                "INDHHIN2", # Total household income
                "WTMEC2YR", "SDMVPSU", "SDMVSTRA")]
demo_vars <- names(demo) # nhanesTableVars('DEMO', 'DEMO_J', namesOnly=TRUE)
demo1 <- nhanesTranslate('DEMO_J', demo_vars, data=demo)
#> No translation table is available for SEQN
#> Translated columns: RIAGENDR DMDBORN4 RIDRETH3 DMDEDUC3 DMDEDUC2 DMDMARTL INDHHIN2
```

NHANES Data Components:

- Demographic (variables like age, gender, income, etc.)
- Blood Pressure (Diastolic and Systolic pressure)
- Body Measures (BMI, Waist Circumference, etc.)
- Smoking Status (Current smoker or not)
- Cholesterol (Total cholesterol in different units)
- Biochemistry Profile (Triglycerides, Uric acid, etc.)
- Physical Activity (Vigorous work and recreational activities)
- Diabetes (Whether the respondent has been told by a doctor that they have diabetes)

Blood pressure component:

```r
bpx <- nhanes('BPX_J')
bpx <- bpx[c("SEQN", # Respondent sequence number
             "BPXDI1", #Diastolic: Blood pres (1st rdg) mm Hg
             "BPXSY1" # Systolic: Blood pres (1st rdg) mm Hg
             )]
bpx_vars <- names(bpx)
bpx1 <- nhanesTranslate('BPX_J', bpx_vars, data=bpx)
#> No translation table is available for SEQN
#> Warning in nhanesTranslate("BPX_J", bpx_vars, data = bpx): No columns were
#> translated
```

Body measure component:

```r
bmi <- nhanes('BMX_J')
bmi <- bmi[c("SEQN", # Respondent sequence number
               "BMXWT", # Weight (kg)
               "BMXHT", # Standing Height (cm)
               "BMXBMI", # Body Mass Index (kg/m**2): 2 YEARS - 150 YEARS
               #"BMDBMIC", # BMI Category - Children/Youth # 2 YEARS - 19 YEARS
```

```
               "BMXWAIST" # Waist Circumference (cm): 2 YEARS - 150 YEARS
               )]
bmi_vars <- names(bmi)
bmi1 <- nhanesTranslate('BMX_J', bmi_vars, data=bmi)
#> No translation table is available for SEQN
#> Warning in nhanesTranslate("BMX_J", bmi_vars, data = bmi): No columns were
#> translated
```

Smoking component:

```
smq <- nhanes('SMQ_J')
smq <- smq[c("SEQN", # Respondent sequence number
               "SMQ040" # Do you now smoke cigarettes?: 18 YEARS - 150 YEARS
               )]
smq_vars <- names(smq)
smq1 <- nhanesTranslate('SMQ_J', smq_vars, data=smq)
#> No translation table is available for SEQN
#> Translated columns: SMQ040
```

```
# alq <- nhanes('ALQ_J')
# alq <- alq[c("SEQN", # Respondent sequence number
#                "ALQ130" # Avg # alcoholic drinks/day - past 12 mos
#                # 18 YEARS - 150 YEARS
#                )]
# alq_vars <- names(alq)
# alq1 <- nhanesTranslate('ALQ_J', alq_vars, data=alq)
```

Cholesterol component:

```
chl <- nhanes('TCHOL_J') # 6 YEARS - 150 YEARS
chl <- chl[c("SEQN", # Respondent sequence number
               "LBXTC", # Total Cholesterol (mg/dL)
               "LBDTCSI" # Total Cholesterol (mmol/L)
               )]
chl_vars <- names(chl)
chl1 <- nhanesTranslate('TCHOL_J', chl_vars, data=chl)
#> No translation table is available for SEQN
#> Warning in nhanesTranslate("TCHOL_J", chl_vars, data = chl): No columns were
#> translated
```

Biochemistry Profile component:

```
tri <- nhanes('BIOPRO_J') # 12 YEARS - 150 YEARS
tri <- tri[c("SEQN", # Respondent sequence number
               "LBXSTR", # Triglycerides, refrig serum (mg/dL)
               "LBXSUA", # Uric acid
               "LBXSTP", # total Protein (g/dL)
               "LBXSTB", # Total Bilirubin (mg/dL)
               "LBXSPH", # Phosphorus (mg/dL)
               "LBXSNASI", # Sodium (mmol/L)
               "LBXSKSI", # Potassium (mmol/L)
               "LBXSGB", # Globulin (g/dL)
               "LBXSCA" # Total Calcium (mg/dL)
               )]
tri_vars <- names(tri)
tri1 <- nhanesTranslate('BIOPRO_J', tri_vars, data=tri)
#> No translation table is available for SEQN
#> Warning in nhanesTranslate("BIOPRO_J", tri_vars, data = tri): No columns were
#> translated
```

Physical activity component:

```
paq <- nhanes('PAQ_J')
paq <- paq[c("SEQN", # Respondent sequence number
               "PAQ605", # Vigorous work activity
               "PAQ650" # Vigorous recreational activities
               )]
paq_vars <- names(paq)
paq1 <- nhanesTranslate('PAQ_J', paq_vars, data=paq)
#> No translation table is available for SEQN
#> Translated columns: PAQ605 PAQ650
```

Diabetes component:

```
diq <- nhanes('DIQ_J')
diq <- diq[c("SEQN", # Respondent sequence number
               "DIQ010" # Doctor told you have diabetes
               )]
diq_vars <- names(diq)
diq1 <- nhanesTranslate('DIQ_J', diq_vars, data=diq)
```

```
#> No translation table is available for SEQN
#> Translated columns: DIQ010
```

Merging all the datasets

> **Tip**
>
> We can use the `merge` or `Reduce` function to combine the datasets

```
analytic.data7 <- Reduce(function(x,y) merge(x,y,by="SEQN",all=TRUE) ,
       list(demo1,bpx1,bmi1,smq1,chl1,tri1,paq1,diq1))
dim(analytic.data7)
#> [1] 9254   33
```

All these datasets are merged into one analytic dataset using the SEQN as the key. This can be done either all at once using the `Reduce` function or one by one (using `merge` once at a time).

```
# Merging one by one
# analytic.data0 <- merge(demo1, bpx1, by = c("SEQN"), all=TRUE)
# analytic.data1 <- merge(analytic.data0, bmi1, by = c("SEQN"), all=TRUE)
# analytic.data2 <- merge(analytic.data1, smq1, by = c("SEQN"), all=TRUE)
# analytic.data3 <- merge(analytic.data2, alq1, by = c("SEQN"), all=TRUE)
# analytic.data4 <- merge(analytic.data3, chl1, by = c("SEQN"), all=TRUE)
# analytic.data5 <- merge(analytic.data4, tri1, by = c("SEQN"), all=TRUE)
# analytic.data6 <- merge(analytic.data5, paq1, by = c("SEQN"), all=TRUE)
# analytic.data7 <- merge(analytic.data6, diq1, by = c("SEQN"), all=TRUE)
# dim(analytic.data7)
```

Check Target population and avoid zero-cell cross-tabulation

See that marital status variable was restricted to 20 YEARS - 150 YEARS.

The dataset is then filtered to only include adults (20 years and older) and avoid zero-cell cross-tabulation.

```
str(analytic.data7)
#> 'data.frame':    9254 obs. of  33 variables:
#>  $ SEQN    : num  93703 93704 93705 93706 93707 ...
#>  $ RIAGENDR: Factor w/ 2 levels "Male","Female": 2 1 2 1 1 2 2 2 1 1 ...
```

```
#>  $ RIDAGEYR: num  2 2 66 18 13 66 75 0 56 18 ...
#>  $ DMDBORN4: Factor w/ 4 levels "Born in 50 US states or Washingt",..: 1 1 1 1 1 2 1 1 2 2
#>  $ RIDRETH3: Factor w/ 6 levels "Mexican American",..: 5 3 4 5 6 5 4 3 5 1 ...
#>  $ DMDEDUC3: Factor w/ 17 levels "Never attended / kindergarten on",..: NA NA NA 16 7 NA NA
#>  $ DMDEDUC2: Factor w/ 7 levels "Less than 9th grade",..: NA NA 2 NA NA 1 4 NA 5 NA ...
#>  $ DMDMARTL: Factor w/ 7 levels "Married","Widowed",..: NA NA 3 NA NA 1 2 NA 1 NA ...
#>  $ INDHHIN2: Factor w/ 16 levels "$ 0 to $ 4,999",..: 14 14 3 NA 10 6 2 14 14 4 ...
#>  $ WTMEC2YR: num  8540 42567 8338 8723 7065 ...
#>  $ SDMVPSU : num  2 1 2 2 1 2 1 1 2 2 ...
#>  $ SDMVSTRA: num  145 143 145 134 138 138 136 134 134 147 ...
#>  $ BPXDI1  : num  NA NA NA 74 38 NA 66 NA 68 68 ...
#>  $ BPXSY1  : num  NA NA NA 112 128 NA 120 NA 108 112 ...
#>  $ BMXWT   : num  13.7 13.9 79.5 66.3 45.4 53.5 88.8 10.2 62.1 58.9 ...
#>  $ BMXHT   : num  88.6 94.2 158.3 175.7 158.4 ...
#>  $ BMXBMI  : num  17.5 15.7 31.7 21.5 18.1 23.7 38.9 NA 21.3 19.7 ...
#>  $ BMXWAIST: num  48.2 50 101.8 79.3 64.1 ...
#>  $ SMQ040  : Factor w/ 3 levels "Every day","Some days",..: NA NA 3 NA NA NA 1 NA NA 2 ...
#>  $ LBXTC   : num  NA NA 157 148 189 209 176 NA 238 182 ...
#>  $ LBDTCSI : num  NA NA 4.06 3.83 4.89 5.4 4.55 NA 6.15 4.71 ...
#>  $ LBXSTR  : num  NA NA 95 92 110 72 132 NA 59 124 ...
#>  $ LBXSUA  : num  NA NA 5.8 8 5.5 4.5 6.2 NA 4.2 5.8 ...
#>  $ LBXSTP  : num  NA NA 7.3 7.1 8 7.1 7 NA 7.1 8.1 ...
#>  $ LBXSTB  : num  NA NA 0.6 0.7 0.7 0.5 0.3 NA 0.3 0.8 ...
#>  $ LBXSPH  : num  NA NA 4 4 4.3 3.3 3.5 NA 3.4 5.1 ...
#>  $ LBXSNASI: num  NA NA 141 144 137 144 141 NA 140 141 ...
#>  $ LBXSKSI : num  NA NA 4 4.4 3.3 4.4 4.1 NA 4.9 4.3 ...
#>  $ LBXSGB  : num  NA NA 2.9 2.7 2.8 3.2 3.3 NA 3.1 3.3 ...
#>  $ LBXSCA  : num  NA NA 9.2 9.6 10.1 9.5 9.9 NA 9.4 9.6 ...
#>  $ PAQ605  : Factor w/ 3 levels "Yes","No","Don't know": NA NA 2 2 NA 2 2 NA 2 1 ...
#>  $ PAQ650  : Factor w/ 2 levels "Yes","No": NA NA 2 2 NA 2 2 NA 1 1 ...
#>  $ DIQ010  : Factor w/ 4 levels "Yes","No","Borderline",..: 2 2 2 2 2 3 2 NA 2 2 ...
head(analytic.data7)
#>    SEQN RIAGENDR RIDAGEYR                          DMDBORN4
#> 1 93703   Female        2 Born in 50 US states or Washingt
#> 2 93704     Male        2 Born in 50 US states or Washingt
#> 3 93705   Female       66 Born in 50 US states or Washingt
#> 4 93706     Male       18 Born in 50 US states or Washingt
#> 5 93707     Male       13 Born in 50 US states or Washingt
#> 6 93708   Female       66                            Others
#>                            RIDRETH3              DMDEDUC3
```

```
#> 1                Non-Hispanic Asian                  <NA>
#> 2                Non-Hispanic White                  <NA>
#> 3                Non-Hispanic Black                  <NA>
#> 4                Non-Hispanic Asian More than high school
#> 5 Other Race - Including Multi-Rac             6th grade
#> 6                Non-Hispanic Asian                  <NA>
#>                           DMDEDUC2 DMDMARTL           INDHHIN2  WTMEC2YR
#> 1                             <NA>     <NA>  $100,000 and Over  8539.731
#> 2                             <NA>     <NA>  $100,000 and Over 42566.615
#> 3 9-11th grade (Includes 12th grad Divorced $10,000 to $14,999  8338.420
#> 4                             <NA>     <NA>               <NA>  8723.440
#> 5                             <NA>     <NA> $65,000 to $74,999  7064.610
#> 6               Less than 9th grade  Married $25,000 to $34,999 14372.489
#>   SDMVPSU SDMVSTRA BPXDI1 BPXSY1 BMXWT BMXHT BMXBMI BMXWAIST     SMQ040 LBXTC
#> 1       2      145     NA     NA  13.7  88.6   17.5     48.2       <NA>    NA
#> 2       1      143     NA     NA  13.9  94.2   15.7     50.0       <NA>    NA
#> 3       2      145     NA     NA  79.5 158.3   31.7    101.8 Not at all   157
#> 4       2      134     74    112  66.3 175.7   21.5     79.3       <NA>   148
#> 5       1      138     38    128  45.4 158.4   18.1     64.1       <NA>   189
#> 6       2      138     NA     NA  53.5 150.2   23.7     88.2       <NA>   209
#>   LBDTCSI LBXSTR LBXSUA LBXSTP LBXSTB LBXSPH LBXSNASI LBXSKSI LBXSGB LBXSCA
#> 1      NA     NA     NA     NA     NA     NA       NA      NA     NA     NA
#> 2      NA     NA     NA     NA     NA     NA       NA      NA     NA     NA
#> 3    4.06     95    5.8    7.3    0.6    4.0      141     4.0    2.9    9.2
#> 4    3.83     92    8.0    7.1    0.7    4.0      144     4.4    2.7    9.6
#> 5    4.89    110    5.5    8.0    0.7    4.3      137     3.3    2.8   10.1
#> 6    5.40     72    4.5    7.1    0.5    3.3      144     4.4    3.2    9.5
#>   PAQ605 PAQ650     DIQ010
#> 1   <NA>   <NA>         No
#> 2   <NA>   <NA>         No
#> 3     No     No         No
#> 4     No     No         No
#> 5   <NA>   <NA>         No
#> 6     No     No Borderline
summary(analytic.data7$RIDAGEYR)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    0.00   11.00   31.00   34.33   58.00   80.00
```

```
dim(analytic.data7)
#> [1] 9254   33
```

```r
analytic.data8 <- analytic.data7
analytic.data8$RIDAGEYR[analytic.data8$RIDAGEYR < 20] <- NA
#analytic.data8 <- subset(analytic.data7, RIDAGEYR >= 20)
dim(analytic.data8)
#> [1] 9254   33
```

Get rid of variables where target was less than 20 years of age accordingly.

```r
analytic.data8$DMDEDUC3 <- NULL # not relevant for adults
#analytic.data8$BMDBMIC <- NULL # not relevant for adults
```

Get rid of invalid responses

Variables that have "Don't Know" or "Refused" as responses are set to NA, effectively getting rid of invalid responses.

```r
factor.names <- c("RIAGENDR","DMDBORN4","RIDRETH3",
                  "DMDEDUC2","DMDMARTL","INDHHIN2",
                  "SMQ040", "PAQ605", "PAQ650", "DIQ010")
numeric.names <- c("SEQN","RIDAGEYR","WTMEC2YR",
                   "SDMVPSU", "SDMVSTRA",
                   "BPXDI1", "BPXSY1", "BMXWT", "BMXHT",
                   "BMXBMI", "BMXWAIST",
                   "ALQ130", "LBXTC", "LBDTCSI",
                   "LBXSTR", "LBXSUA", "LBXSTP", "LBXSTB",
                   "LBXSPH", "LBXSNASI", "LBXSKSI",
                   "LBXSGB","LBXSCA")
analytic.data8[factor.names] <- apply(X = analytic.data8[factor.names],
                                      MARGIN = 2, FUN = as.factor)
# analytic.data8[numeric.names] <- apply(X = analytic.data8[numeric.names],
#                                        MARGIN = 2, FUN =
#                                          function (x) as.numeric(as.character(x)))
```

```r
analytic.data9 <- analytic.data8
analytic.data9$DMDBORN4[analytic.data9$DMDBORN4 == "Don't Know"] <- NA
#analytic.data9 <- subset(analytic.data8, DMDBORN4 != "Don't Know")
dim(analytic.data9)
#> [1] 9254   32

```

```r
analytic.data10 <- analytic.data9
analytic.data10$DMDEDUC2[analytic.data10$DMDEDUC2 == "Don't Know"] <- NA
#analytic.data10 <- subset(analytic.data9, DMDEDUC2 != "Don't Know")
dim(analytic.data10)
#> [1] 9254   32

analytic.data11 <- analytic.data10
analytic.data11$DMDMARTL[analytic.data11$DMDMARTL == "Don't Know"] <- NA
analytic.data11$DMDMARTL[analytic.data11$DMDMARTL == "Refused"] <- NA
# analytic.data11 <- subset(analytic.data10, DMDMARTL != "Don't Know" & DMDMARTL != "Refused")
dim(analytic.data11)
#> [1] 9254   32


analytic.data12 <- analytic.data11
analytic.data12$INDHHIN2[analytic.data12$INDHHIN2 == "Don't Know"] <- NA
analytic.data12$INDHHIN2[analytic.data12$INDHHIN2 == "Refused"] <- NA
analytic.data12$INDHHIN2[analytic.data12$INDHHIN2 == "Under $20,000"] <- NA
analytic.data12$INDHHIN2[analytic.data12$INDHHIN2 == "$20,000 and Over"] <- NA
# analytic.data12 <- subset(analytic.data11, INDHHIN2 != "Don't know" & INDHHIN2 !=  "Refused"
dim(analytic.data12)
#> [1] 9254   32

#analytic.data11 <- subset(analytic.data10, ALQ130 != 777 & ALQ130 != 999 )
#dim(analytic.data11) # this are listed as NA anyway

analytic.data13 <- analytic.data12
analytic.data13$PAQ605[analytic.data13$PAQ605 == "Don't know"] <- NA
analytic.data13$PAQ605[analytic.data13$PAQ605 == "Refused"] <- NA
# analytic.data13 <- subset(analytic.data12, PAQ605 != "Don't know" & PAQ605 != "Refused")
dim(analytic.data13)
#> [1] 9254   32

analytic.data14 <- analytic.data13
analytic.data14$PAQ650[analytic.data14$PAQ650 == "Don't know"] <- NA
analytic.data14$PAQ650[analytic.data14$PAQ650 == "Refused"] <- NA
# analytic.data14 <- subset(analytic.data13, PAQ650 != "Don't Know" & PAQ650 != "Refused")
dim(analytic.data14)
#> [1] 9254   32

```

```r
analytic.data15 <- analytic.data14
analytic.data15$DIQ010[analytic.data15$DIQ010 == "Don't know"] <- NA
analytic.data15$DIQ010[analytic.data15$DIQ010 == "Refused"] <- NA
# analytic.data15 <- subset(analytic.data14, DIQ010 != "Don't Know" & DIQ010 != "Refused")
dim(analytic.data15)
#> [1] 9254   32


# analytic.data15$ALQ130[analytic.data15$ALQ130 > 100] <- NA
# summary(analytic.data15$ALQ130)
table(analytic.data15$SMQ040,useNA = "always")
#>
#>  Every day Not at all  Some days       <NA>
#>        805       1338        216       6895
table(analytic.data15$PAQ605,useNA = "always")
#>
#>   No  Yes <NA>
#> 4461 1389 3404
table(analytic.data15$PAQ650,useNA = "always")
#>
#>   No  Yes <NA>
#> 4422 1434 3398
table(analytic.data15$PAQ650,useNA = "always")
#>
#>   No  Yes <NA>
#> 4422 1434 3398
```

Recode values

Let us recode the variables using the `recode` function:

```r
require(car)
#> Loading required package: car
#> Loading required package: carData
analytic.data15$RIDRETH3 <- recode(analytic.data15$RIDRETH3,
                            "c('Mexican American','Other Hispanic')='Hispanic';
                            'Non-Hispanic White'='White';
                            'Non-Hispanic Black'='Black';
                            c('Non-Hispanic Asian',
```

```
                                 'Other Race - Including Multi-Rac')='Other';
                                 else=NA")
analytic.data15$DMDEDUC2 <- recode(analytic.data15$DMDEDUC2,
                              "c('Some college or AA degree',
                               'College graduate or above')='College';
                              c('9-11th grade (Includes 12th grad',
                                'High school graduate/GED or equi')
                                 ='High.School';
                              'Less than 9th grade'='School';
                                 else=NA")
analytic.data15$DMDMARTL <- recode(analytic.data15$DMDMARTL,
                              "c('Divorced','Separated','Widowed')
                                  ='Previously.married';
                              c('Living with partner', 'Married')
                                  ='Married';
                              'Never married'='Never.married';
                                 else=NA")
analytic.data15$INDHHIN2 <- recode(analytic.data15$INDHHIN2,
                              "c('$ 0 to $ 4,999', '$ 5,000 to $ 9,999',
                                   '$10,000 to $14,999', '$15,000 to $19,999',
                                   '$20,000 to $24,999')='<25k';
                              c('$25,000 to $34,999', '$35,000 to $44,999',
                                   '$45,000 to $54,999') = 'Between.25kto54k';
                              c('$55,000 to $64,999', '$65,000 to $74,999',
                                   '$75,000 to $99,999')='Between.55kto99k';
                              '$100,000 and Over'= 'Over100k';
                                 else=NA")
analytic.data15$SMQ040 <- recode(analytic.data15$SMQ040,
                              "'Every day'='Every.day';
                              'Not at all'='Not.at.all';
                              'Some days'='Some.days';
                                 else=NA")
analytic.data15$DIQ010 <- recode(analytic.data15$DIQ010,
                              "'No'='No';
                              c('Yes', 'Borderline')='Yes';
                                 else=NA")
```

Data types for various variables are set correctly; for instance, factor variables are converted to factor data types, and numeric variables to numeric data types.

Check missingness

> **Tip**
>
> We can use the `plot_missing` function to plot the profile of missing values, e.g., the percentage of missing per variable

```
require(DataExplorer)
#> Loading required package: DataExplorer
plot_missing(analytic.data15)
```

A subsequent chapter will delve into the additional factors that impact how we handle missing data.

Check data summaries

```
names(analytic.data15)
#>  [1] "SEQN"     "RIAGENDR" "RIDAGEYR" "DMDBORN4" "RIDRETH3" "DMDEDUC2"
#>  [7] "DMDMARTL" "INDHHIN2" "WTMEC2YR" "SDMVPSU"  "SDMVSTRA" "BPXDI1"
#> [13] "BPXSY1"   "BMXWT"    "BMXHT"    "BMXBMI"   "BMXWAIST" "SMQ040"
#> [19] "LBXTC"    "LBDTCSI"  "LBXSTR"   "LBXSUA"   "LBXSTP"   "LBXSTB"
#> [25] "LBXSPH"   "LBXSNASI" "LBXSKSI"  "LBXSGB"   "LBXSCA"   "PAQ605"
#> [31] "PAQ650"   "DIQ010"
```

```r
names(analytic.data15) <- c("ID", "gender", "age", "born", "race", "education",
"married", "income", "weight", "psu", "strata", "diastolicBP",
"systolicBP", "bodyweight", "bodyheight", "bmi", "waist", "smoke",
"cholesterol", "cholesterolM2", "triglycerides",
"uric.acid", "protein", "bilirubin", "phosphorus", "sodium",
"potassium", "globulin", "calcium", "physical.work",
"physical.recreational","diabetes")
require("tableone")
#> Loading required package: tableone
CreateTableOne(data = analytic.data15, includeNA = TRUE)
#>
#>                                        Overall
#>   n                                        9254
#>   ID (mean (SD))                       98329.50 (2671.54)
#>   gender = Male (%)                        4557 (49.2)
#>   age (mean (SD))                         51.50 (17.81)
#>   born (%)
#>      Born in 50 US states or Washingt     7303 (78.9)
#>      Others                               1948 (21.1)
#>      Refused                                 2 ( 0.0)
#>      NA                                      1 ( 0.0)
#>   race (%)
#>      Black                                2115 (22.9)
#>      Hispanic                             2187 (23.6)
#>      Other                                1802 (19.5)
#>      White                                3150 (34.0)
#>   education (%)
#>      College                              3114 (33.7)
#>      High.School                          1963 (21.2)
#>      School                                479 ( 5.2)
#>      NA                                   3698 (40.0)
#>   married (%)
#>      Married                              3252 (35.1)
#>      Never.married                        1006 (10.9)
#>      Previously.married                   1305 (14.1)
#>      NA                                   3691 (39.9)
#>   income (%)
#>      <25k                                 1998 (21.6)
#>      Between.25kto54k                     2460 (26.6)
#>      Between.55kto99k                     1843 (19.9)
```

```
#>      Over100k                              1624 (17.5)
#>      NA                                    1329 (14.4)
#>   weight (mean (SD))                   34670.71 (43344.00)
#>   psu (mean (SD))                          1.52 (0.50)
#>   strata (mean (SD))                     140.97 (4.20)
#>   diastolicBP (mean (SD))                 67.84 (16.36)
#>   systolicBP (mean (SD))                 121.33 (19.98)
#>   bodyweight (mean (SD))                  65.14 (32.89)
#>   bodyheight (mean (SD))                 156.59 (22.26)
#>   bmi (mean (SD))                         26.58 (8.26)
#>   waist (mean (SD))                       89.93 (22.81)
#>   smoke (%)
#>      Every.day                             805 ( 8.7)
#>      Not.at.all                           1338 (14.5)
#>      Some.days                             216 ( 2.3)
#>      NA                                   6895 (74.5)
#>   cholesterol (mean (SD))                179.89 (40.60)
#>   cholesterolM2 (mean (SD))                4.65 (1.05)
#>   triglycerides (mean (SD))              137.44 (109.13)
#>   uric.acid (mean (SD))                    5.40 (1.48)
#>   protein (mean (SD))                      7.17 (0.44)
#>   bilirubin (mean (SD))                    0.46 (0.28)
#>   phosphorus (mean (SD))                   3.66 (0.59)
#>   sodium (mean (SD))                     140.32 (2.75)
#>   potassium (mean (SD))                    4.09 (0.36)
#>   globulin (mean (SD))                     3.09 (0.43)
#>   calcium (mean (SD))                      9.32 (0.37)
#>   physical.work (%)
#>      No                                   4461 (48.2)
#>      Yes                                  1389 (15.0)
#>      NA                                   3404 (36.8)
#>   physical.recreational (%)
#>      No                                   4422 (47.8)
#>      Yes                                  1434 (15.5)
#>      NA                                   3398 (36.7)
#>   diabetes (%)
#>      No                                   7816 (84.5)
#>      Yes                                  1077 (11.6)
#>      NA                                    361 ( 3.9)
```

Create complete case data (for now)

```
analytic.with.miss <- analytic.data15
analytic.with.miss$cholesterol.bin <- ifelse(analytic.with.miss$cholesterol <200, 1,0)
analytic <- as.data.frame(na.omit(analytic.with.miss))
dim(analytic)
#> [1] 1562   33
```

Creating Table 1 from the complete case data

```
require("tableone")
CreateTableOne(data = analytic, includeNA = TRUE)
#>
#>                                    Overall
#>   n                                    1562
#>   ID (mean (SD))                   98344.21 (2697.76)
#>   gender = Male (%)                     959 (61.4)
#>   age (mean (SD))                    53.18 (17.18)
#>   born = Others (%)                     299 (19.1)
#>   race (%)
#>      Black                              324 (20.7)
#>      Hispanic                           284 (18.2)
#>      Other                              228 (14.6)
#>      White                              726 (46.5)
#>   education (%)
#>      College                            806 (51.6)
#>      High.School                        658 (42.1)
#>      School                              98 ( 6.3)
#>   married (%)
#>      Married                            921 (59.0)
#>      Never.married                      228 (14.6)
#>      Previously.married                 413 (26.4)
#>   income (%)
#>      <25k                               484 (31.0)
#>      Between.25kto54k                   520 (33.3)
#>      Between.55kto99k                   331 (21.2)
#>      Over100k                           227 (14.5)
#>   weight (mean (SD))               48538.53 (54106.24)
```

```
#>   psu (mean (SD))                    1.48 (0.50)
#>   strata (mean (SD))               141.18 (4.07)
#>   diastolicBP (mean (SD))           72.06 (14.17)
#>   systolicBP (mean (SD))           127.06 (19.11)
#>   bodyweight (mean (SD))            85.66 (22.41)
#>   bodyheight (mean (SD))           168.96 (9.30)
#>   bmi (mean (SD))                   29.96 (7.33)
#>   waist (mean (SD))                102.98 (17.15)
#>   smoke (%)
#>      Every.day                       530 (33.9)
#>      Not.at.all                      903 (57.8)
#>      Some.days                       129 ( 8.3)
#>   cholesterol (mean (SD))          188.77 (43.51)
#>   cholesterolM2 (mean (SD))          4.88 (1.13)
#>   triglycerides (mean (SD))        154.71 (123.00)
#>   uric.acid (mean (SD))              5.62 (1.53)
#>   protein (mean (SD))                7.09 (0.43)
#>   bilirubin (mean (SD))              0.46 (0.27)
#>   phosphorus (mean (SD))             3.53 (0.54)
#>   sodium (mean (SD))               140.14 (2.83)
#>   potassium (mean (SD))              4.10 (0.38)
#>   globulin (mean (SD))               3.03 (0.44)
#>   calcium (mean (SD))                9.29 (0.37)
#>   physical.work = Yes (%)             476 (30.5)
#>   physical.recreational = Yes (%)     290 (18.6)
#>   diabetes = Yes (%)                  330 (21.1)
#>   cholesterol.bin (mean (SD))        0.63 (0.48)
```

Saving data

Additional factors come into play when dealing with complex survey datasets; these will be explored in a subsequent chapter.

```
# getwd()
save(analytic.with.miss, analytic, file="Data/researchquestion/NHANES17.RData")
```

References

R functions (Q)

The list of new R functions introduced in this *Research question* lab component are below:

Function_name	Package_name	Use
as.data.frame	base	To force an object to a data frame
as.formula	base/stats	To specify a model formula, e.g., formula for an outcome model
confint	base/stats	To estimate the confidence interval for model parameters
degf	survey	To see the degrees of freedom for a survey design object
describe	DescTools	To see the summary statistics of variables
exp	base	Exponentials
glm	base/stats	To run generalized linear models
lapply	base	To apply a function over a list, e.g., to see the summary of a list of variables
length	base	To see the length of an object, e.g., number of elements/observations of a var
plot_missing	DataExplorer	To plot the profile of missing values, e.g., the percentage of missing per varia
publish	Publish	To show/publish regression tables
Reduce	base	To combine multiple objects, e.g., datasets
round	base	To round numeric values
saveRDS	base	To save a single R object. Similarly, readDRS will read an R object
skim	skimr	To see the summary statistics of variables
svydesign	survey	To create a design for the survey data analysis
svyglm	survey	To run design-adjusted generalized linear models
unique	base	To see the number of unique elements
weights	base/stats	To extract model weights, e.g., see the weights from a pre-specified survey de

For more information, visit the resources mentioned earlier.

Quiz (Q)

> ⚠ Warning
>
> We are currently in the beta testing phase for our quiz page updates, so please be aware that some responses may not be accurate. Thank you for your patience.
> For those interested in downloading the quiz questions, you'll find a section below dedicated to downloading.

Live Quiz

You can click on the live quiz options to either see hints of why your response may be incorrect (response marked in red), or whether your choice of response is correct (response marked in green):

Download Quiz

1. **Downloading the File:**
   - Navigate to the link: See here.
   - Right-click on the link and select "Save link as…" from the dropdown menu.
   - Alternatively click here for downloading the quizzes.
   - Choose a destination folder on your computer where you'd like to save the file (e.g., Desktop). **Remember this location**, as you'll need to navigate to it later.
2. **Setting Up RStudio:**

- If you don't have RStudio installed, see the download link in here.
- Launch RStudio after installing.

3. **Installing Necessary R Packages:**

   - Before running the R Markdown file, ensure you have the required packages. In RStudio's console (located at the bottom left by default), enter the following commands to install them:

```
install.packages("learnr")
install.packages("xfun")
```

4. **Opening and Running the File:**

   - In RStudio, go to `File` > `Open File...`.
   - Navigate to the folder where you saved the `Rmd` file and select it to open.
   - Once the file is open in RStudio, you'll see a "Run Document" button (green) at the top of the script editor. Click on it to run the R Markdown Quiz file.

App (Q)

Below is an example of an app that utilizes the NHANES dataset from the prediction question tutorial. Users can generate regression outcomes and generate a Table 1 from it based on selected predictors.

> Tip
>
> **Web App may need some time to load the necessary files, depending on your internet speed.**

> Important
>
> If you want to run the app locally in your computer instead for faster load time: you can download the app directly in your R via `shiny::runGitHub("EpiShinyLiveQ", "ehsanx", subdir = "app")`. Make sure that `shiny`, `Publish`, `jtools`, `ggstance`, `broom.mixed`, `huxtable`, `httr` and `tableone` packages are installed in your computer/R.

> Warning
>
> **Bug Report**:
> Fill out this form to report any issues with the app.

Part IV

Causal roles

Background

This chapter delves deep into the intricate issues surrounding **causal associations**, particularly confounding, mediation, and other related biases. In this comprehensive series of tutorials, various aspects of confounding and bias are explored through the lens of Directed Acyclic Graphs (DAGs). Initially, the tutorials guide you through the process of generating large datasets based on these DAGs. They then delve into how the inclusion of different types of variables in adjustment models can skew estimates of treatment effects. We use the R package `simcausal` in these tutorials to derive empirical estimates from a large dataset.

> **!** Important
>
> **Datasets**:
> We use the R package `simcausal` in these tutorials to derive empirical estimates from a large simulated dataset. The simulation is based on data generation based on specified DAGs.

In the preceding chapter, we delved into the various types of research questions, distinguishing between causal and predictive inquiries. This current chapter focuses primarily on the challenges and intricacies associated with causal questions. Specifically, we will explore which types of variables are most appropriate to incorporate into adjustment models when aiming to estimate treatment effects accurately. In contrast, the subsequent chapter will shift its emphasis towards predictive questions, providing insights into their unique characteristics and considerations.

Directed Acyclic Graphs (DAGs) are powerful tools in the realm of causal inference (see the concepts page). They are a type of graphical representation used to depict causal relationships between variables. This visual representation of the causal structure among variables makes it easier to understand and communicate complex causal relationships. They provide a visual framework to understand, represent, and analyze complex causal relationships, ensuring that researchers make informed decisions when trying to answer causal questions.

Overview of tutorials

Confounding

The first tutorial provides a thorough exploration of confounding, with a particular focus on its impact on treatment effect estimates in large datasets. It emphasizes the importance of properly adjusting for confounders to arrive at accurate estimates.

Mediator

This tutorial focuses on the role of mediator variables in estimating treatment effects. It assesses how adjusting for the me-

diator influences the estimated treatment effect, exploring both scenarios where the true treatment effect is either non-null or null.

Collider

This tutorial serves as a practical guide for understanding how the inclusion of colliders can affect the estimation of treatment effects in causal models.

Z-bias

This tutorial explores the concept of Z-bias, a phenomenon that can lead to misleading estimates of treatment effects in observational studies. It demonstrates how failing to properly adjust or not adjust for instrumental variables can result in biased estimates and compares these with the true treatment effect.

Collapsibility

This tutorial provides a detailed guide on calculating marginal probabilities and measures of association, including Risk Difference (RD), Risk Ratio (RR), and Odds Ratio (OR). It examines the impact of adjusting for various covariates on these measures, highlighting the concept of "collapsibility."

Change-in-estimate

This tutorial focuses on the "Change-in-estimate" concept to understand the impact of various variables on measures of effect. For both continuous and binary outcomes, the tutorial reveals that adding a confounder to the model alters the true treatment effect estimate. Conversely, including a variable that is not a confounder but is a pure risk factor can either change or not change the effect estimate, depending on the type of outcome involved. This nuanced approach aids in understanding how different roles of variables can influence results and interpretations in causal inference.

Tip

Optional Content:
You'll find that some sections conclude with an optional video walkthrough that demonstrates the code. Keep in mind that the content might have been updated since these videos were recorded. Watching these videos is optional.

Warning

Bug Report:
Fill out this form to report any issues with the tutorial.

Concepts (R)

Confounding

Confounding is a pervasive concern in epidemiology, especially in observational studies focusing on causality. Epidemiologists need to carefully select confounders to avoid biased results due to third factors affecting the relationship between exposure and outcome. Commonly used methods for selecting confounders, such as change-in-estimator or solely relying on p-value-based statistical methods, may be inadequate or even problematic.

Epidemiologists need a more formalized system for confounder selection, incorporating causal diagrams and counterfactual reasoning. This includes an understanding of the underlying causal relationships and the potential impacts of different variables on the observed association. Understanding the temporal order and causal pathways is crucial for accurate confounder control.

However, it is possible that epidemiologists may lack comprehensive knowledge about the causal roles of all variables and hence may need to resort to empirical criteria such as the disjunctive cause criterion, or other variable selection methods such as machine learning approaches. While these methods can provide more sophisticated analyses and help address the high dimensionality and complex structures of modern epidemiological data, epidemiologists need to understand how these approaches function, along with their benefits and limitations, to avoid introducing additional bias into the analysis.

Effect modifier

Effect modification and **interaction** are two distinct concepts in epidemiology. Effect modification occurs when the causal effect of an exposure (A) on an outcome (Y) varies based on the levels of a third factor (B).

In this scenario, the association between the exposure and the outcome differs within the strata of a second exposure, which acts as the effect modifier. For instance, the impact of alcohol (A) on oral cancer (Y) might differ based on tobacco smoking (B).

On the other hand, interaction refers to the joint causal effect of two exposures (A and B) on an outcome (Y). It examines how the combination of multiple exposures influences the outcome, such as the combined effect of alcohol (A) and tobacco smoking (B) on oral cancer (Y).

In essence, while effect modification looks at how a third factor influences the relationship between an exposure and an outcome, interaction focuses on the combined effect of two exposures on the outcome.

Table 2 fallacy

The "Table 2 Fallacy" in epidemiology refers to the misleading practice of presenting multiple adjusted effect estimates from a single statistical model in one table, often resulting in misinterpretation. This occurs when researchers report both the primary exposure's effects and secondary exposures' (often an adjustment variable for the primary exposure) effects without adequately distinguishing between the types of effects or considering the causal relationships among variables.

This idea highlights the potential for misunderstanding in interpreting the effects of various exposures on an outcome when they are reported together, leading to confusion over the nature and magnitude of the relationships and possibly influencing the design and interpretation of further studies (Westreich and Greenland 2013). The fallacy demonstrates the need

for careful consideration of the types of effects estimated and reported in statistical models, urging researchers to be clear about the distinctions and implications of controlled direct effects, total effects, and the presence of confounding or mediating variables.

Reading list

Confounding key reference: (VanderWeele 2019)

Effect modification key reference: (VanderWeele 2009)

Table 2 fallacy key reference: (Westreich and Greenland 2013)

Optional reading:

- (Tennant et al. 2021)
- (Wright 1921)
- (Greenland, Pearl, and Robins 1999a)
- (Lederer et al. 2019)
- (Etminan, Collins, and Mansournia 2020)
- (Heinze, Wallisch, and Dunkler 2018)

Video Lessons

Potential outcome framework

What is included in this Video Lesson:

- 0:00 Introduction
- 0:16 Notations
- 2:40 Treatment Effect
- 6:13 Real-world Problem of the counterfactual definition
- 9:44 Real-world Solution in Observational Setting

The timestamps are also included in the YouTube video description.

DAG

The video lesson split into 3 parts

DAG codes:

Example DAG codes can be accessed from this GitHub repository folder

Empirical criteria

When complete knowledge of DAG is unavailable

Modelling criteria for variable selection

Most of these modelling criteria work when we are only dealing with confounders (some important ones, and some less so), or maybe risk factors of the outcomes. But no mediators, or colliders.

Effect modification vs. interaction

Important

To revisit or deepen your grasp of these two concepts, consider reviewing this external tutorial.

Table 2 fallacy

Video Lesson Slides

Confounding

Effect modification

Table 2 fallacy

Links

Confounding

- Google Slides
- PDF Slides

Effect modification

- Google Slides
- PDF Slides

Table 2 fallacy

- Google Slides
- PDF Slides
- External link from dagitty

References

Confounding

This tutorial aims to delve into the role of confounding variables in data analysis, especially in the context of big data. We will examine each of these using simulations built on Directed Acyclic Graphs (DAGs). The objective is to understand whether a simple regression adjusting for the confounder variable can correctly estimate treatment effects in such a large sample.

```r
# devtools::install_github('osofr/simcausal')
require(simcausal)
```

Big data: **What if we had 1,000,000 (one million) observations?** Would that give us true result? Let's try to answer that using DAGs.

Let us consider

- L is continuous variable
- A is binary treatment
- Y is continuous outcome

An example of A could be receiving the right heart catheterization (RHC) procedure or not, Y could be the length of hospital stay, and L could be age (see here).

Non-null effect

- True treatment effect = 1.3

Data generating process

To perform the lab, we'll need the `simcausal` R package. This package may not be available on CRAN but can be installed from the author's GitHub page.

```
require(simcausal)
D <- DAG.empty()
D <- D +
  node("L", distr = "rnorm", mean = 10, sd = 1) +
  node("A", distr = "rbern", prob = plogis(-10 + 1.1*L)) +
  node("Y", distr = "rnorm", mean = 0.5 * L + 1.3 * A, sd = 0.1)
Dset <- set.DAG(D)
```

Note that `rnorm` and `rbern` generate random samples from Normal and Bernoulli distributions, respectively. In the above code chuck, we set the parameters to generate L from Normal distribution with a mean of 10 and a standard deviation of 1. Similarly, we set the parameters to generate A from the Bernoulli distribution with the probability of logit of (-10 + 1.1 × L). Finally, we set the parameters to generate Y from the Normal distribution with a mean of (0.5 × L + 1.3 × A) and a standard deviation of 0.1.

Generate DAG

Let us draw a directed acyclic graph (DAG). Below we use the `plotDAG` function from the `simcausal` package. However, we can draw a DAG using DAGitty.

```
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
#> using the following vertex attributes:
#> 120.8NAdarkbluenone0
#> using the following edge attributes:
#> 0.50.40.7black1
```

As per the DAG, L is a confounder. When exploring the relationship between A and Y, we need to adjust our model for L to get an unbiased estimate of A on Y.

Generate Data

Now, let us simulate the data using the defined parameters and the DAG. Below, we generate data for 1,000,000 participants. We set a seed 123 so that one can reproduce the same dataset.

```
require(simcausal)
Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
head(Obs.Data)
#>   ID         L A        Y
#> 1  1  9.439524 1 6.165097
#> 2  2  9.769823 1 6.040117
#> 3  3 11.558708 1 7.121178
#> 4  4 10.070508 0 4.943313
#> 5  5 10.129288 1 6.438245
#> 6  6 11.715065 1 7.073990
```

Estimate effect

Now we will fit the generalized linear model (glm), without and with adjusting for L.

```
# Not adjusted for L
fit0 <- glm(Y ~ A, family="gaussian", data=Obs.Data)
round(coef(fit0),2)
#> (Intercept)           A
#>        4.69        1.75

# Adjusted for L
fit <- glm(Y ~ A + L, family="gaussian", data=Obs.Data)
round(coef(fit),2)
#> (Intercept)           A           L
#>         0.0         1.3         0.5
```

> **!** Important
>
> In this case, our true treatment effect is 1.3. When we estimate the relationship between A and Y without adjusting for L, we obtain an estimated effect of 1.75. However, this is not the true effect. The true treatment effect of 1.3 is recovered when we adjust for L.

Null effect

- True treatment effect = 0

Let us see the results when there is no treatment effect, i.s., the true treatment effect is zero.

Data generating process

```
require(simcausal)
D <- DAG.empty()
D <- D +
  node("L", distr = "rnorm", mean = 10, sd = 1) +
```

```r
  node("A", distr = "rbern", prob = plogis(-10 + 1.1*L)) +
  node("Y", distr = "rnorm", mean = 0.5 * L + 0 * A, sd = .1)
Dset <- set.DAG(D)
```

Generate DAG

```r
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
```

Generate Data

```r
require(simcausal)
Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
head(Obs.Data)
#>   ID         L A        Y
#> 1  1  9.439524 1 4.865097
#> 2  2  9.769823 1 4.740117
#> 3  3 11.558708 1 5.821178
#> 4  4 10.070508 0 4.943313
#> 5  5 10.129288 1 5.138245
#> 6  6 11.715065 1 5.773990
```

Estimate effect

```r
# Not adjusted for L
fit0 <- glm(Y ~ A, family = "gaussian", data = Obs.Data)
round(coef(fit0),2)
#> (Intercept)           A
#>        4.69        0.45

# Adjusted for L
fit <- glm(Y ~ A + L, family = "gaussian", data = Obs.Data)
round(coef(fit),2)
#> (Intercept)           A           L
#>         0.0         0.0         0.5
```

> **Important**
>
> In this second scenario, the true treatment effect is zero. There is no arrow from A to Y in the DAG, but L remains a common cause for both. Upon analyzing the data without adjusting for L, we observe an induced correlation between A and Y. This correlation disappears, confirming the true null effect, when we adjust for L.

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Mediator

Mediators play a crucial role in understanding how a treatment variable affects an outcome. A mediator variable lies in the pathway between the treatment and outcome, essentially transmitting or explaining the effect of the treatment variable. In this expanded tutorial, we'll delve into more details based on the lecture, specifically focusing on the true direct and indirect effects when a mediator is present.

```r
# Load required packages
library(simcausal)
```

Let us consider

- M is continuous variable
- A is binary treatment
- Y is continuous outcome

An example of A could be receiving the right heart catheterization (RHC) procedure or not, Y could be the length of hospital stay, and M could be the number of comorbidities.

Non-null effect

- True treatment effect = 1.3

Our true treatment effect is 1.3, and the mediator variable's effect on the outcome Y is 0.5. It's important to differentiate between these effects.

Data generating process

```r
require(simcausal)
D <- DAG.empty()
D <- D +
  node("A", distr = "rbern", prob = plogis(-10)) +
  node("M", distr = "rnorm", mean = 10 + 0.9 * A, sd = 1) +
  node("Y", distr = "rnorm", mean = 0.5 * M + 1.3 * A, sd = .1)
Dset <- set.DAG(D)
```

Generate DAG

```r
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
```

As we can see, M is a mediator (mediates the effect of A on Y). When exploring the total effect of A on Y, we should not adjust our model for M.

Generate Data

```r
require(simcausal)
Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
head(Obs.Data)
#>   ID A         M        Y
#> 1  1 0 10.262000 5.276335
#> 2  2 0  8.479041 4.094726
#> 3  3 0 10.236628 5.160138
#> 4  4 0 10.660304 5.238211
#> 5  5 0  9.785994 4.966598
#> 6  6 0 10.192827 5.012871
```

Estimate effect

```r
# Not adjusted for M
fit0 <- glm(Y ~ A, family="gaussian", data=Obs.Data)
round(coef(fit0),2)
#> (Intercept)           A
#>        5.00        1.69

# Adjusted for M
fit <- glm(Y ~ A + M, family="gaussian", data=Obs.Data)
round(coef(fit),2)
#> (Intercept)           A           M
#>         0.0         1.3         0.5
```

> **Important**
>
> You might notice a total effect that could differ from the true effects. In the lecture example, a crude association showed an effect of 1.69, which is the total effect combining both direct and indirect pathways.
> Upon adjusting for M, the coefficients will show you the direct effect of A on Y and the indirect effect through M. These should align closely with our true effects: a direct effect of 1.3 and an indirect effect of 0.5.
> In this expanded tutorial, we've shown how essential it is to consider mediator variables when estimating treatment effects. We've also illustrated how adjusting for media-

tors allows you to differentiate between true direct and indirect effects, thereby reducing the risk of drawing incorrect conclusions from your data.
Detailed on the mediation analysis can be found in the Mediation analysis chapter.

Null effect

- True treatment effect = 0

Data generating process

```
require(simcausal)
D <- DAG.empty()
D <- D +
  node("A", distr = "rbern", prob = plogis(-10)) +
  node("M", distr = "rnorm", mean = 10 + 0.9 * A, sd = 1) +
  node("Y", distr = "rnorm", mean = 0.5 * M, sd = .1)
Dset <- set.DAG(D)
```

Generate DAG

```
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
```

Generate Data

```
require(simcausal)
Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
head(Obs.Data)
#>   ID A         M        Y
#> 1  1 0 10.262000 5.276335
#> 2  2 0  8.479041 4.094726
#> 3  3 0 10.236628 5.160138
#> 4  4 0 10.660304 5.238211
#> 5  5 0  9.785994 4.966598
#> 6  6 0 10.192827 5.012871
```

Estimate effect

```
# Not adjusted for M
fit0 <- glm(Y ~ A, family="gaussian", data=Obs.Data)
round(coef(fit0),2)
#> (Intercept)           A
#>        5.00        0.39

# Adjusted for M
```

```r
fit <- glm(Y ~ A + M, family="gaussian", data=Obs.Data)
round(coef(fit),2)
#> (Intercept)           A           M
#>         0.0         0.0         0.5
```

> **Important**
>
> **Total Effect**: If you want to measure the "total effect" of a treatment on an outcome, then you typically don't adjust for the mediator. The reason is that the total effect captures both the direct effect of the treatment on the outcome and the indirect effect through the mediator.
> **Direct and Indirect Effects**: If you want to separate out the direct and indirect effects, then you would adjust for the mediator. In essence, when you control for the mediator, what remains is the direct effect of the treatment on the outcome.
> **Linearity and Decomposition**: In linear models with continuous outcomes, it is more straightforward to decompose total effects into direct and indirect effects. The mathematics get more complicated in non-linear models or when dealing with non-continuous outcomes.

Collider

In causal inference, understanding the role of colliders is crucial. A collider is a variable that is a common effect of two or more variables. Adjusting for a collider can introduce bias into your estimates.

```r
# Load required packages
library(simcausal)
```

In a DAG, a collider is a variable influenced by two or more other variables. In our case, L is a collider because it is affected by both A (the treatment) and Y (the outcome). When you adjust for a collider like L, you could introduce bias into your estimates, as demonstrated in the examples below.

Let us consider

- L is continuous variable
- A is binary treatment/exposure
- Y is continuous outcome

An example of A could be a genetic variable (e.g., skin color), Y could be an environmental variable (e.g., indoor air pollution), and L could be disease conditions (e.g., number of comorbidities) (detailed expamles here).

Non-null effect

- True treatment effect = 1.3

Data generating process

```r
D <- DAG.empty()
D <- D +
  node("A", distr = "rbern", prob = plogis(-10)) +
  node("Y", distr = "rnorm", mean = 1.3 * A, sd = .1) +
  node("L", distr = "rnorm", mean = 10 * Y + 1.3 * A, sd = 1)
Dset <- set.DAG(D)
```

Generate DAG

```r
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
```

Generate data

```r
require(simcausal)
Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
head(Obs.Data)
#>   ID A           Y          L
#> 1  1 0  0.02620001  1.7153509
```

```
#> 2  2 0 -0.15209594 -2.9689005
#> 3  3 0  0.02366280  0.6548666
#> 4  4 0  0.06603038 -0.2591076
#> 5  5 0 -0.02140063  0.5220096
#> 6  6 0  0.01928270 -0.6426011
```

Estimate effect

```r
# Not adjusted for L
fit0 <- glm(Y ~ A, family="gaussian", data=Obs.Data)
round(coef(fit0),2)
#> (Intercept)           A
#>        0.00        1.29

# Adjusted for L
fit <- glm(Y ~ A + L, family="gaussian", data=Obs.Data)
round(coef(fit),2)
#> (Intercept)           A           L
#>        0.00        0.58        0.05
```

> **❗ Important**
>
> When not adjusting for L, we recover the true effect close to 1.3. Adjusting for L introduces bias, making the estimate unreliable.

Null effect

- True treatment effect = 0

Data generating process

```r
D <- DAG.empty()
D <- D +
  node("A", distr = "rbern", prob = plogis(-10)) +
```

```r
  node("Y", distr = "rnorm", mean = 0, sd = .1) +
  node("L", distr = "rnorm", mean = 10 * Y + 1.3 * A, sd = 1)
Dset <- set.DAG(D)
```

Generate DAG

```r
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
```

Generate data

```r
require(simcausal)
Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
head(Obs.Data)
#>   ID A           Y          L
#> 1  1 0  0.02620001  1.7153509
#> 2  2 0 -0.15209594 -2.9689005
#> 3  3 0  0.02366280  0.6548666
#> 4  4 0  0.06603038 -0.2591076
```

```
#> 5  5 0 -0.02140063  0.5220096
#> 6  6 0  0.01928270 -0.6426011
```

Estimate effect

```r
# Not adjusted for L
fit0 <- glm(Y ~ A, family="gaussian", data=Obs.Data)
round(coef(fit0),2)
#> (Intercept)           A
#>        0.00       -0.01

# Adjusted for L
fit <- glm(Y ~ A + L, family="gaussian", data=Obs.Data)
round(coef(fit),2)
#> (Intercept)           A           L
#>        0.00       -0.07        0.05
```

> **Important**
>
> When the true effect is null, not adjusting for L shows an estimate close to zero. Adjusting for L moves the estimate away from the null value, introducing bias.

Even 1,000,000 observations were not enough to recover true treatment effect! But we are close enough.

Z-bias

Z-bias occurs in the context of causal inference, specifically when using instrumental variables to estimate causal effects. Instrumental variables (IVs) are used to isolate the variation in the treatment variable that is unrelated to the confounding factors, thus providing a pathway to estimate causal effects.

```r
# Load required packages
library(simcausal)
```

Continuous Y

- U is unmeasured continuous variable
- Z is an instrumental variable
- A is binary treatment
- Y is continuous outcome

Non-null effect

- True treatment effect = 1.3

0.0.0.0.1 * Data generating process

```r
require(simcausal)
D <- DAG.empty()
D <- D +
  node("U", distr = "rnorm", mean = 2, sd = 1) +
  node("Z", distr = "rnorm", mean = 2, sd = 1) +
  node("A", distr = "rbern", prob = plogis(-1 + 2*U + 2*Z)) +
  node("Y", distr = "rnorm", mean = -1 + 3 * U + 1.3 * A, sd = 0.1)
Dset <- set.DAG(D)
```

0.0.0.0.2 * Generate DAG

```
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
```

0.0.0.0.3 * Generate Data

```
require(simcausal)
Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
head(Obs.Data)
#>   ID        U         Z A         Y
#> 1  1 1.439524 0.9919293 1  4.667744
#> 2  2 1.769823 3.3549394 1  5.491603
#> 3  3 3.558708 1.5310251 1 11.063519
#> 4  4 2.070508 3.4681936 1  6.501764
#> 5  5 2.129288 2.4425564 1  6.834678
#> 6  6 3.715065 2.1462031 1 11.340150
```

0.0.0.0.4 * Estimate effect

```
# True data generating mechanism (unattainable as U is unmeasured)
fit0 <- glm(Y ~ A + U, family="gaussian", data=Obs.Data)
```

```r
round(coef(fit0),2)
#> (Intercept)           A           U
#>        -1.0         1.3         3.0

# Unadjusted effect (Z not controlled)
fit1 <- glm(Y ~ A, family="gaussian", data=Obs.Data)
round(coef(fit1),2)
#> (Intercept)           A
#>        0.79        5.59

# Bias fit 1
coef(fit1)["A"] - 1.3
#>      A
#> 4.2935

# Adjusted effect (Z  controlled)
fit2 <- glm(Y ~ A + Z, family="gaussian", data=Obs.Data)
round(coef(fit2),2)
#> (Intercept)           A           Z
#>        0.86        5.77       -0.12

# Bias from fit 2
coef(fit2)["A"] - 1.3
#>        A
#> 4.465787
```

Binary Y

- U is unmeasured continuous variable
- Z is an instrumental variable
- A is binary treatment
- Y is binary outcome

Non-null effect

- True treatment effect = 1.3

0.0.0.0.1 * Data generating process

```
require(simcausal)
D <- DAG.empty()
D <- D +
  node("U", distr = "rnorm", mean = 2, sd = 1) +
  node("Z", distr = "rnorm", mean = 2, sd = 1) +
  node("A", distr = "rbern", prob = plogis(-1 + 2*U + 2*Z)) +
  node("Y", distr = "rbern", prob = plogis(-1 + 3 * U + 1.3 * A))
Dset <- set.DAG(D)
```

0.0.0.0.2 * Generate DAG

```
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
```

0.0.0.0.3 * Generate Data

```
require(simcausal)
Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
head(Obs.Data)
#>   ID        U         Z A Y
```

```
#> 1  1 1.439524 0.9919293 1 1
#> 2  2 1.769823 3.3549394 1 1
#> 3  3 3.558708 1.5310251 1 1
#> 4  4 2.070508 3.4681936 1 1
#> 5  5 2.129288 2.4425564 1 1
#> 6  6 3.715065 2.1462031 1 1
```

0.0.0.0.4 * Estimate effect

```
# True data generating mechanism (unattainable as U is unmeasured)
fit0 <- glm(Y ~ A + U, family="binomial", data=Obs.Data)
round(coef(fit0),2)
#> (Intercept)           A           U
#>       -0.99        1.30        3.01

# Unadjusted effect (Z not controlled)
fit1 <- glm(Y ~ A, family="binomial", data=Obs.Data)
round(coef(fit1),2)
#> (Intercept)           A
#>        0.40        3.02

# Bias fit 1
coef(fit1)["A"] - 1.3
#>        A
#> 1.716482

# Adjusted effect (Z  controlled)
fit2 <- glm(Y ~ A + Z, family="binomial", data=Obs.Data)
round(coef(fit2),2)
#> (Intercept)           A           Z
#>        0.51        3.29       -0.18

# Bias from fit 2
coef(fit2)["A"] - 1.3
#>        A
#> 1.991396
```

Collapsibility

Let us assume we have a regression of hypertension (Y), smoking (A) and a risk factor for outcome, gender (L). Then let us set up 2 regression models:

- 1st regression model is $Y \sim \beta \times A + \alpha \times L$. Here we are conditioning on gender (L).
- 2nd regression model is $Y \sim \beta' \times A$

Then regression is collapsible for β over L if $\beta = \beta'$ from the 2nd regression omitting L. $\beta \neq \beta'$ would mean non-collapsibility. A measure of association (say, risk difference) is collapsible if the marginal measure of association is equal to a weighted average of the stratum-specific measures of association. Non-collapsibility is also knows as Simpson's Paradox (in absence of confounding of course): a statistical phenomenon where an association between two factors (say, hypertension and smoking) in a population (we are talking about marginal estimate here) is different than the associations of same relationship in sub-populations (conditional on some other factor, say, age; hence talking about conditional estimates). Risk ratio and risk difference are collapsible.

Odds ratio can be non-collapsible. It can produce different treatment effect estimate for different covariate adjustment sets (see example below of when adjusting form age and sex vs. when adjusting none). This is true even in the absence of confounding. However, according to our definition here, OR is collapsible when we consider gender in the adjustment set.

Note that, OR non-collapsibility is a consequence of the fact that it is estimated via a logit link function (nonlinearity of the logistic transformation).

Ref:

- (Greenland, Pearl, and Robins 1999b)
- (Mansournia and Greenland 2015)

```r
# Load required packages
library(simcausal)
library(tableone)
library(Publish)
library(lawstat)
```

Data generating process

Let us generate the data with smoking being the exposure and hypertension being the outcome.

```r
require(simcausal)
D <- DAG.empty()
D <- D +
  node("gender", distr = "rbern", prob = 0.7) +
  node("age", distr = "rnorm", mean = 2, sd = 4) +
  node("smoking", distr = "rbern", prob = plogis(.1)) +
  node("hypertension", distr = "rbern",
       prob = plogis(1 + log(3.5) * smoking + log(.1) * gender + log(7) * age))
Dset <- set.DAG(D)
```

Generate DAG

```r
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
```

Generate Data

```
Obs.Data <- sim(DAG = Dset, n = 100000, rndseed = 123)
head(Obs.Data)
#>   ID gender        age smoking hypertension
#> 1  1      1  3.0383589       0            1
#> 2  2      0  5.6700429       1            1
#> 3  3      1 -0.8892734       0            0
#> 4  4      0 -1.2331361       1            1
#> 5  5      0  1.4345919       0            1
#> 6  6      1 11.0280538       1            1
```

Balance check

```
require(tableone)
CreateTableOne(data = Obs.Data, strata = "smoking", vars = c("gender", "age"))
#>                      Stratified by smoking
#>                       0            1            p      test
#>   n                   47720        52280
#>   gender (mean (SD))  0.70 (0.46)  0.70 (0.46)  0.403
#>   age (mean (SD))     2.02 (4.02)  2.01 (4.00)  0.690
```

Conditional and crude risk difference (RD)

Since RD is a collapsible measure, the marginal and conditional estimate should be approximately the same.

Full list of risk factors for outcome (2 variables)

```
## RD
require(Publish)
fitx0 <- glm(hypertension ~ smoking + gender + age,
             family=gaussian(link = "identity"), data=Obs.Data)
publish(fitx0, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[2,]
#>   Variable Units Coefficient       CI.95 p-value
#> 2  smoking              0.06 [0.05;0.06] < 1e-04
```

Ref:

- (Naimi and Whitcomb 2020) ("For the risk difference, one may use a GLM with a Gaussian (i.e., normal) distribution and identity link function, or, equivalently, an ordinary least squares estimator …robust variance estimator (or bootstrap) should be used to obtain valid standard errors.")

Strtatum specific (2 variables)

```
fitx1 <- glm(hypertension ~ smoking, family=gaussian(link = "identity"),
             data=subset(Obs.Data, gender == 1 & age < 2))
publish(fitx1, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[2,]
#>   Variable Units Coefficient       CI.95 p-value
#> 2  smoking              0.11 [0.10;0.12] < 1e-04

fitx2 <- glm(hypertension ~ smoking, family=gaussian(link = "identity"),
             data=subset(Obs.Data, gender == 0 & age < 2))
publish(fitx2, print = FALSE, confint.method = "robust",
```

```
        pvalue.method = "robust")$regressionTable[2,]
#>   Variable Units Coefficient       CI.95 p-value
#> 2  smoking              0.10 [0.08;0.12] < 1e-04

fitx3 <- glm(hypertension ~ smoking, family=gaussian(link = "identity"),
             data=subset(Obs.Data, gender == 1 & age >= 2))
publish(fitx3, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[2,]
#>   Variable Units Coefficient       CI.95 p-value
#> 2  smoking              0.01 [0.00;0.01] < 1e-04

fitx4 <- glm(hypertension ~ smoking, family=gaussian(link = "identity"),
             data=subset(Obs.Data, gender == 0 & age >= 2))
publish(fitx4, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[2,]
#>   Variable Units Coefficient        CI.95  p-value
#> 2  smoking              0.00 [-0.00;0.00]   0.3754
```

Unweighted mean from all strata (for simplicity, 2 variables)

```
round(mean(c(coef(fitx1)["smoking"],
             coef(fitx2)["smoking"],
             coef(fitx3)["smoking"],
             coef(fitx4)["smoking"])),2)
#> [1] 0.05
```

Partial list of risk factors for outcome (1 variable)

```
fitx0 <- glm(hypertension ~ smoking + gender,
             family=gaussian(link = "identity"), data=Obs.Data)
publish(fitx0, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[2,]
#>   Variable Units Coefficient       CI.95 p-value
#> 2  smoking              0.05 [0.05;0.06] < 1e-04
```

Strtatum specific (1 variable)

```
fitx1 <- glm(hypertension ~ smoking, family=gaussian(link = "identity"),
             data=subset(Obs.Data, gender == 1))
publish(fitx1, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[2,]
#>   Variable Units Coefficient       CI.95 p-value
#> 2  smoking              0.06 [0.05;0.07] < 1e-04

fitx2 <- glm(hypertension ~ smoking, family=gaussian(link = "identity"),
             data=subset(Obs.Data, gender == 0))
publish(fitx2, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[2,]
#>   Variable Units Coefficient       CI.95 p-value
#> 2  smoking              0.05 [0.04;0.06] < 1e-04
```

Unweighted mean from all strata (for simplicity, 1 variable)

```
round(mean(c(coef(fitx1)["smoking"],
             coef(fitx2)["smoking"])),2)
#> [1] 0.05
```

Crude (in absence of confounding)

```
fitx0 <- glm(hypertension ~ smoking,
             family=gaussian(link = "identity"), data=Obs.Data)
publish(fitx0, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[2,]
#>   Variable Units Coefficient       CI.95 p-value
#> 2  smoking              0.05 [0.05;0.06] < 1e-04
```

Conditional and crude risk ratio (RR)

Ref:

- (Naimi and Whitcomb 2020) ("For the risk ratio, one may use a GLM with a Poisson distribution and log link function .... one should use the robust (or sandwich) variance estimator to obtain valid standard errors (the bootstrap can also be used)").

Full list of risk factors for outcome (2 variables)

```
fitx0 <- glm(hypertension ~ smoking + gender + age,
             family=poisson(link = "log"), data=Obs.Data)
publish(fitx0, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[1,]
#>   Variable Units HazardRatio       CI.95 p-value
#> 1  smoking              1.08 [1.08;1.09] < 1e-04
```

The estimate we see as a `hazard ratio` is basically a `risk ratio`.

Strtatum specific (2 variables)

```
fitx1 <- glm(hypertension ~ smoking,
             family=poisson(link = "log"),
             data=subset(Obs.Data, gender == 1 & age < 2))
publish(fitx1, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[1,]
#>   Variable Units HazardRatio       CI.95 p-value
#> 1  smoking              1.41 [1.36;1.45] < 1e-04

fitx2 <- glm(hypertension ~ smoking,
             family=poisson(link = "log"),
             data=subset(Obs.Data, gender == 0 & age < 2))
publish(fitx2, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[1,]
#>   Variable Units HazardRatio       CI.95 p-value
#> 1  smoking              1.21 [1.17;1.25] < 1e-04

fitx3 <- glm(hypertension ~ smoking,
```

```r
             family=poisson(link = "log"),
             data=subset(Obs.Data, gender == 1 & age >= 2))
publish(fitx3, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[1,]
#>   Variable Units HazardRatio       CI.95 p-value
#> 1  smoking              1.01 [1.00;1.01] < 1e-04

fitx4 <- glm(hypertension ~ smoking,
             family=poisson(link = "log"),
             data=subset(Obs.Data, gender == 0 & age >= 2))
publish(fitx4, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[1,]
#>   Variable Units HazardRatio       CI.95  p-value
#> 1  smoking              1.00 [1.00;1.00]   0.3754
```

Unweighted mean from all strata (for simplicity, 2 variables)

```r
mean(exp(c(coef(fitx1)["smoking"],
           coef(fitx2)["smoking"],
           coef(fitx3)["smoking"],
           coef(fitx4)["smoking"])))
#> [1] 1.156387
```

Partial list of risk factors for outcome (1 variable)

```r
fitx0 <- glm(hypertension ~ smoking + gender,
             family=poisson(link = "log"), data=Obs.Data)
publish(fitx0, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[1,]
#>   Variable Units HazardRatio       CI.95 p-value
#> 1  smoking              1.08 [1.07;1.09] < 1e-04
```

Strtatum specific (1 variable)

```r
fitx1 <- glm(hypertension ~ smoking,
             family=poisson(link = "log"),
             data=subset(Obs.Data, gender == 1))
publish(fitx1, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[1,]
#>   Variable Units HazardRatio       CI.95 p-value
#> 1  smoking              1.09 [1.08;1.10] < 1e-04
fitx2 <- glm(hypertension ~ smoking,
             family=poisson(link = "log"),
             data=subset(Obs.Data, gender == 0))
publish(fitx2, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[1,]
#>   Variable Units HazardRatio       CI.95 p-value
#> 1  smoking              1.06 [1.05;1.08] < 1e-04
```

Unweighted mean from all strata (for simplicity, 1 variable)

```r
mean(exp(c(coef(fitx1)["smoking"],
           coef(fitx2)["smoking"])))
#> [1] 1.077402
```

Crude (in absence of confounding)

```r
fitx0 <- glm(hypertension ~ smoking,
             family=poisson(link = "log"), data=Obs.Data)
publish(fitx0, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[1,]
#>   Variable Units HazardRatio       CI.95 p-value
#> 1  smoking              1.08 [1.07;1.09] < 1e-04
```

Since the marginal and conditional estimate are approximately the same, RD is a collapsible measure.

Conditional and crude OR

Full list of risk factors for outcome (2 variables)

```
fitx0 <- glm(hypertension ~ smoking + gender + age, family=binomial(link = "logit"), data=Obs.
publish(fitx0, print = FALSE, confint.method = "robust", pvalue.method = "robust")$regressionTa
#>   Variable Units OddsRatio       CI.95 p-value
#> 1  smoking            3.37 [3.17;3.58] < 1e-04
```

Strtatum specific (2 variables)

```
fitx1 <- glm(hypertension ~ smoking,
             family=binomial(link = "logit"),
             data=subset(Obs.Data, gender == 1 & age < 2))
publish(fitx1, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[1,]
#>   Variable Units OddsRatio       CI.95 p-value
#> 1  smoking            1.65 [1.57;1.73] < 1e-04

fitx2 <- glm(hypertension ~ smoking,
             family=binomial(link = "logit"),
             data=subset(Obs.Data, gender == 0 & age < 2))
publish(fitx2, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[1,]
#>   Variable Units OddsRatio       CI.95 p-value
#> 1  smoking            1.49 [1.40;1.59] < 1e-04

fitx3 <- glm(hypertension ~ smoking,
             family=binomial(link = "logit"),
             data=subset(Obs.Data, gender == 1 & age >= 2))
publish(fitx3, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[1,]
#>   Variable Units OddsRatio       CI.95 p-value
#> 1  smoking            3.45 [2.49;4.77] < 1e-04

fitx4 <- glm(hypertension ~ smoking,
             family=binomial(link = "logit"),
             data=subset(Obs.Data, gender == 0 & age >= 2))
```

```r
publish(fitx4, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[1,]
#>   Variable Units OddsRatio        CI.95  p-value
#> 1  smoking            2.14 [0.39;11.66]   0.3811
```

Unweighted mean from all strata (for simplicity, 2 variables)

```r
mean(exp(c(coef(fitx1)["smoking"],
           coef(fitx2)["smoking"],
           coef(fitx3)["smoking"],
           coef(fitx4)["smoking"])))
#> [1] 2.180804
```

Partial list of risk factors for outcome (1 variable)

```r
fitx0 <- glm(hypertension ~ smoking + gender,
             family=binomial(link = "logit"), data=Obs.Data)
publish(fitx0, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[1,]
#>   Variable Units OddsRatio       CI.95 p-value
#> 1  smoking            1.29 [1.26;1.33] < 1e-04
```

Strtatum specific (1 variable)

```r
fitx1 <- glm(hypertension ~ smoking,
             family=binomial(link = "logit"),
             data=subset(Obs.Data, gender == 1))
publish(fitx1, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[1,]
#>   Variable Units OddsRatio       CI.95 p-value
#> 1  smoking            1.29 [1.25;1.34] < 1e-04

fitx2 <- glm(hypertension ~ smoking,
             family=binomial(link = "logit"),
             data=subset(Obs.Data, gender == 0))
publish(fitx2, print = FALSE, confint.method = "robust",
```

```
        pvalue.method = "robust")$regressionTable[1,]
#>   Variable Units OddsRatio       CI.95 p-value
#> 1  smoking            1.29 [1.22;1.36] < 1e-04
```

Unweighted mean from all strata (for simplicity, 1 variable)

```
mean(exp(c(coef(fitx1)["smoking"],
           coef(fitx2)["smoking"])))
#> [1] 1.290429
```

Mantel-Haenszel adjusted ORs with 1 variable

```
tabx <- xtabs( ~ hypertension + smoking + gender, data = Obs.Data)
ftable(tabx)
#>                       gender     0     1
#> hypertension smoking
#> 0            0                3788 12401
#>              1                3400 11504
#> 1            0               10547 20984
#>              1               12178 25198
# require(samplesizeCMH)
# apply(tabx, 3, odds.ratio)

library(lawstat)
cmh.test(tabx)
#>
#>  Cochran-Mantel-Haenszel Chi-square Test
#>
#> data:  tabx
#> CMH statistic = NA, df = 1.0000, p-value = NA, MH Estimate = 1.2924,
#> Pooled Odd Ratio = 1.2876, Odd Ratio of level 1 = 1.2864, Odd Ratio of
#> level 2 = 1.2945
# mantelhaen.test(tabx, exact = TRUE)
```

Crude (in absence of confounding)

```
fitx0 <- glm(hypertension ~ smoking,
             family=binomial(link = "logit"), data=Obs.Data)
publish(fitx0, print = FALSE, confint.method = "robust",
        pvalue.method = "robust")$regressionTable[1,]
#>   Variable Units OddsRatio       CI.95 p-value
#> 1  smoking            1.29 [1.25;1.32] < 1e-04
```

Marginal RD, RR and OR

Below we show a procedure for calculating marginal probabilities p_1 (for treated) and p_0 (for untreated).

Adjustment of 2 variables

```
fitx3 <- glm(hypertension ~ smoking + gender + age,
             family=binomial(link = "logit"), data=Obs.Data)
Obs.Data.all.tx <- Obs.Data
Obs.Data.all.tx$smoking <- 1
p1 <- mean(predict(fitx3,
                   newdata = Obs.Data.all.tx, type = "response"))
Obs.Data.all.utx <- Obs.Data
Obs.Data.all.utx$smoking <- 0
p0 <- mean(predict(fitx3,
                   newdata = Obs.Data.all.utx, type = "response"))

RDm <- p1 - p0
RRm <- p1 / p0
ORm <- (p1 / (1-p1)) / (p0 / (1-p0))
ORc <- as.numeric(exp(coef(fitx3)["smoking"]))
RRz <- ORm / ((1-p0) + p0 * ORm) # Zhang and Yu (1998)
cat("RD marginal = ", round(RDm,2),
    "\nRR marginal = ", round(RRm,2),
    "\nOR marginal = ", round(ORm,2),
    "\nOR conditional = ", round(ORc,2),
    "\nRR (ZY)= ", round(RRz,2))
```

```
#> RD marginal =  0.05
#> RR marginal =  1.08
#> OR marginal =  1.29
#> OR conditional =  3.37
#> RR (ZY)=  1.08
```

Adjustment of 1 variable

```
fitx2 <- glm(hypertension ~ smoking + gender,
             family=binomial(link = "logit"), data=Obs.Data)
Obs.Data.all.tx <- Obs.Data
Obs.Data.all.tx$smoking <- 1
p1 <- mean(predict(fitx2,
                   newdata = Obs.Data.all.tx, type = "response"))
Obs.Data.all.utx <- Obs.Data
Obs.Data.all.utx$smoking <- 0
p0 <- mean(predict(fitx2,
                   newdata = Obs.Data.all.utx, type = "response"))

RDm <- p1 - p0
RRm <- p1 / p0
ORm <- (p1 / (1-p1)) / (p0 / (1-p0))
ORc <- as.numeric(exp(coef(fitx2)["smoking"]))
RRz <- ORm / ((1-p0) + p0 * ORm) # Zhang and Yu (1998)
cat("RD marginal = ", round(RDm,2),
    "\nRR marginal = ", round(RRm,2),
    "\nOR marginal = ", round(ORm,2),
    "\nOR conditional = ", round(ORc,2),
    "\nRR (ZY)= ", round(RRz,2))
#> RD marginal =  0.05
#> RR marginal =  1.08
#> OR marginal =  1.29
#> OR conditional =  1.29
#> RR (ZY)=  1.08
```

No adjustment

```r
fitx1 <- glm(hypertension ~ smoking,
             family=binomial(link = "logit"), data=Obs.Data)
Obs.Data.all.tx <- Obs.Data
Obs.Data.all.tx$smoking <- 1
p1 <- mean(predict(fitx0,
                   newdata = Obs.Data.all.tx, type = "response"))
Obs.Data.all.utx <- Obs.Data
Obs.Data.all.utx$smoking <- 0
p0 <- mean(predict(fitx0,
                   newdata = Obs.Data.all.utx, type = "response"))

RDm <- p1 - p0
RRm <- p1 / p0
ORm <- (p1 / (1-p1)) / (p0 / (1-p0))
ORc <- as.numeric(exp(coef(fitx1)["smoking"]))
RRz <- ORm / ((1-p0) + p0 * ORm) # Zhang and Yu (1998)
cat("RD marginal = ", round(RDm,2),
    "\nRR marginal = ", round(RRm,2),
    "\nOR marginal = ", round(ORm,2),
    "\nOR conditional = ", round(ORc,2),
    "\nRR (ZY)= ", round(RRz,2))
#> RD marginal =  0.05
#> RR marginal =  1.08
#> OR marginal =  1.29
#> OR conditional =  1.29
#> RR (ZY)=  1.08
```

Bootstrap could be used to estimate confidence intervals.

Ref:

- (Kleinman and Norton 2009) ("this paper demonstrates how to move from a nonlinear model to estimates of marginal effects that are quantified as the adjusted risk ratio or adjusted risk difference")
- (Peter C. Austin 2010) ("clinically meaningful measures of treatment effect using logistic regression model")
- (Luijken et al. 2022) ("marginal odds ratio")

- (Muller and MacLehose 2014) ("marginal standardization")
- (Greenland 2004) ("standardized / population-averaged")
- (Bieler et al. 2010) ("standardized /population-averaged risk from the logistic model")

Summary

Here are the summary of the results based on a scenario where confounding was absent:

Modelling strategy	RD (conditional)	RR (conditional)	OR (conditional)
age + gender in regression	0.06 [0.05;0.06]	1.08 [1.08;1.09]	3.37 [3.17;3.58]
stratified by age and gender (mean)	0.05 (*0.11, 0.1,0.01,0*)	1.16 (*1.41, 1.21, 1.01,1*)	2.18 (unweighted; *1.65, 1.49, 3.45, 2.14*)
gender in regression	0.05 [0.05;0.06]	1.08 [1.07;1.09]	1.29 [1.26;1.33]
stratified by gender (mean)	0.05 (*0.6,0.5*)	1.08 (*1.09, 1.06*)	1.29 (*1.29, 1.29*; **M-H** 1.29)
Marginal estimates			
crude	0.05 [0.05;0.06]	1.08 [1.07;1.09]	1.29 [1.25;1.32]
Based on marginal probabilities (any variable combination)	0.05	1.08	1.29

As we can see, marginal estimate = conditional estimate for RD and RR but not for OR. Hence, RD and RR are collapsible measures, while OR is a non-collapsible measure.

Video content (optional)

> **Tip**
>
> For those who prefer a video walk through, feel free to watch the video below, which offers a description of an earlier version of the above content.

References

Change-in-estimate

The change-in-estimate criterion is one of the purposeful variable selection criteria, where a relative change is used to determine whether a variable should be included in the model or omitted from the model. This change can be measured by using the relative change in the regression coefficient or relative change by adjusting the effect of the variance estimate. Eliminating a variable using this technique can bias the estimates. In particular, this method is not appropriate when the estimate is non-collapsible. In this chapter, we will see the use of change-in-estimate for various measures of effects (RD and OR, and impact of collapsibility vs non-collapsibility in absence of confounding).

For a collapsible measure (e.g., beta coefficient from a linear model with a continuous outcome, risk difference or risk ratio from a generalized linear model with a binary outcome), our estimate could be the same when we adjust our model or not for a variable that is not a confounder. However, a non-collapsible measure (e.g., odds ratio) could be different whether or not we adjust our model for a variable that is not a confounder.

```r
# Load required packages
library(simcausal)
```

Adjusting for a variable that is a confounder

Continuous Outcome

- True treatment effect = 1.3

Data generating process

```r
require(simcausal)
D <- DAG.empty()
D <- D +
  node("L", distr = "rnorm", mean = 0, sd = 1) +
  node("A", distr = "rnorm", mean = 0 + L, sd = 1) +
  node("P", distr = "rbern", prob = plogis(-10)) +
  node("Y", distr = "rnorm", mean = 1.1 * L + 1.3 * A, sd = .1)
Dset <- set.DAG(D)
```

Generate DAG

```r
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
```

Generate Data

```r
require(simcausal)
Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
head(Obs.Data)
#>   ID           L          A P         Y
#> 1  1 -0.56047565 -1.5685464 0 -2.606463
#> 2  2 -0.23017749  1.1247619 0  1.091131
#> 3  3  1.55870831  1.0897334 0  3.218627
#> 4  4  0.07050839  1.5387020 0  2.068111
#> 5  5  0.12928774  0.5718442 0  1.032429
#> 6  6  1.71506499  1.8612681 0  4.201175
```

Estimate effect (beta-coef)

```r
fit <- glm(Y ~ A, family="gaussian", data=Obs.Data)
round(coef(fit),2)
#> (Intercept)           A
#>        0.00        1.85

fit2 <- glm(Y ~ A + L, family="gaussian", data=Obs.Data)
round(coef(fit2),2)
#> (Intercept)           A           L
#>         0.0         1.3         1.1
```

Including a variable that is a confounder (L) in the model changes effect estimate (1.3).

Binary Outcome

- True treatment effect = 1.3

Data generating process

```r
require(simcausal)
D <- DAG.empty()
D <- D +
```

```r
  node("L", distr = "rnorm", mean = 0, sd = 1) +
  node("A", distr = "rnorm", mean = 0 + L, sd = 1) +
  node("P", distr = "rbern", prob = plogis(-10)) +
  # node("M", distr = "rnorm", mean = P + 0.5 * A, sd = 1) +
  node("Y", distr = "rbern", prob = plogis( 1.1 * L + 1.3 * A))
Dset <- set.DAG(D)
#> ...automatically assigning order attribute to some nodes...
#> node L, order:1
#> node A, order:2
#> node P, order:3
#> node Y, order:4
```

Generate DAG

```r
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
#> using the following vertex attributes:
#> 120.8NAdarkbluenone0
#> using the following edge attributes:
#> 0.50.40.7black1
```

Generate Data

```
require(simcausal)
Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
head(Obs.Data)
#>   ID           L          A P Y
#> 1  1 -0.56047565 -1.5685464 0 0
#> 2  2 -0.23017749  1.1247619 0 1
#> 3  3  1.55870831  1.0897334 0 1
#> 4  4  0.07050839  1.5387020 0 1
#> 5  5  0.12928774  0.5718442 0 0
#> 6  6  1.71506499  1.8612681 0 1
```

Estimate effect (OR)

```
fit <- glm(Y ~ A, family=binomial(link = "logit"), data=Obs.Data)
round(coef(fit),2)
#> (Intercept)           A
#>        0.00        1.68

fit2 <- glm(Y ~ A + L, family=binomial(link = "logit"), data=Obs.Data)
round(coef(fit2),2)
#> (Intercept)           A           L
#>         0.0         1.3         1.1
```

Including a variable that is a confounder (L) in the model changes effect estimate (1.3).

Adjusting for a variable that is not a confounder (simplified)

Continuous Outcome

- True treatment effect = 1.3

Data generating process

```r
require(simcausal)
D <- DAG.empty()
D <- D +
    node("A", distr = "rnorm", mean = 0, sd = 1) +
  node("P", distr = "rbern", prob = plogis(-10)) +
  # node("M", distr = "rnorm", mean = P + 0.5 * A, sd = 1) +
  node("R", distr = "rnorm", mean = 0, sd = 1) +
  node("Y", distr = "rnorm", mean = 1.1 * R + 1.3 * A, sd = .1)
Dset <- set.DAG(D)
#> ...automatically assigning order attribute to some nodes...
#> node A, order:1
#> node P, order:2
#> node R, order:3
#> node Y, order:4
```

Generate DAG

```r
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
#> using the following vertex attributes:
#> 120.8NAdarkbluenone0
#> using the following edge attributes:
#> 0.50.40.7black1
```

Generate Data

```
require(simcausal)
Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
head(Obs.Data)
#>   ID           A P          R          Y
#> 1  1 -0.56047565 0  1.4533508  0.9192381
#> 2  2 -0.23017749 0 -1.4479411 -2.0098302
#> 3  3  1.55870831 0  0.4182387  2.5737773
#> 4  4  0.07050839 0 -0.9194114 -0.9294528
#> 5  5  0.12928774 0  0.7360159  1.1245064
#> 6  6  1.71506499 0 -0.8354281  1.2055689
```

Estimate effect (beta-coef)

```
fit <- glm(Y ~ A, family="gaussian", data=Obs.Data)
round(coef(fit),2)
#> (Intercept)           A
#>         0.0         1.3

fit2 <- glm(Y ~ A + R, family="gaussian", data=Obs.Data)
round(coef(fit2),2)
```

```
#> (Intercept)           A           R
#>         0.0         1.3         1.1
```

Including a variable that is not a confounder (R is a pure risk factor for the outcome Y) in the model does not change effect estimate (1.3).

Binary Outcome

- True treatment effect = 1.3

Data generating process

```
require(simcausal)
D <- DAG.empty()
D <- D +
  node("A", distr = "rnorm", mean = 0, sd = 1) +
  node("P", distr = "rbern", prob = plogis(-10)) +
  # node("M", distr = "rnorm", mean = P + 0.5 * A, sd = 1) +
  node("R", distr = "rnorm", mean = 0, sd = 1) +
  node("Y", distr = "rbern", prob = plogis(1.1 * R + 1.3 * A))
Dset <- set.DAG(D)
#> ...automatically assigning order attribute to some nodes...
#> node A, order:1
#> node P, order:2
#> node R, order:3
#> node Y, order:4
```

Generate DAG

```
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
#> using the following vertex attributes:
#> 120.8NAdarkbluenone0
```

```
#> using the following edge attributes:
#> 0.50.40.7black1
```

Generate Data

```
require(simcausal)
Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
head(Obs.Data)
#>   ID           A P          R Y
#> 1  1 -0.56047565 0  1.4533508 1
#> 2  2 -0.23017749 0 -1.4479411 0
#> 3  3  1.55870831 0  0.4182387 1
#> 4  4  0.07050839 0 -0.9194114 1
#> 5  5  0.12928774 0  0.7360159 0
#> 6  6  1.71506499 0 -0.8354281 0
```

Estimate effect (OR)

```
fit <- glm(Y ~ A, family=binomial(link = "logit"), data=Obs.Data)
round(coef(fit),2)
#> (Intercept)           A
```

```
#>        0.00        1.06

fit2 <- glm(Y ~ A + R, family=binomial(link = "logit"), data=Obs.Data)
round(coef(fit2),2)
#> (Intercept)           A           R
#>         0.0         1.3         1.1
```

Including a variable that is not a confounder (R is a pure risk factor for the outcome Y) in the model changes effect estimate (1.3).

Adjusting for a variable that is not a confounder (Complex)

Continuous Outcome

- True treatment effect = 1.3

Data generating process

```
require(simcausal)
D <- DAG.empty()
D <- D +
    node("A", distr = "rnorm", mean = 0, sd = 1) +
  node("P", distr = "rbern", prob = plogis(-10)) +
  node("M", distr = "rnorm", mean = P + 0.5 * A, sd = 1) +
  node("R", distr = "rnorm", mean = 0, sd = 1) +
  node("Y", distr = "rnorm", mean = 0.5 * M + 1.1 * R + 1.3 * A, sd = .1)
Dset <- set.DAG(D)
#> ...automatically assigning order attribute to some nodes...
#> node A, order:1
#> node P, order:2
#> node M, order:3
#> node R, order:4
#> node Y, order:5
```

Generate DAG

```
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
#> using the following vertex attributes:
#> 120.8NAdarkbluenone0
#> using the following edge attributes:
#> 0.50.40.7black1
```

Generate Data

```
require(simcausal)
Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
head(Obs.Data)
#>   ID           A P           M           R          Y
#> 1  1 -0.56047565 0  1.17311294  0.49170637  0.4103007
#> 2  2 -0.23017749 0 -1.56302981 -1.17864299 -2.3797737
#> 3  3  1.55870831 0  1.19759283  0.87393920  3.5823197
#> 4  4  0.07050839 0 -0.88415718 -0.09761158 -0.5322484
#> 5  5  0.12928774 0  0.80065978  1.46814830  2.2160066
#> 6  6  1.71506499 0  0.02210438 -1.05044636  1.1270274
```

Estimate effect (beta-coef)

```
fit <- glm(Y ~ A + M, family="gaussian", data=Obs.Data)
round(coef(fit),2)
#> (Intercept)           A           M
#>         0.0         1.3         0.5

fit2 <- glm(Y ~ A + M + R, family="gaussian", data=Obs.Data)
round(coef(fit2),2)
#> (Intercept)           A           M           R
#>         0.0         1.3         0.5         1.1
```

Including a variable that is not a confounder (R is a pure risk factor for the outcome Y) in the model does not change effect estimate (1.3).

Binary Outcome

- True treatment effect = 1.3

Data generating process

```
require(simcausal)
D <- DAG.empty()
D <- D +
  node("A", distr = "rnorm", mean = 0, sd = 1) +
  node("P", distr = "rbern", prob = plogis(-10)) +
  node("M", distr = "rnorm", mean = P + 0.5 * A, sd = 1) +
  node("R", distr = "rnorm", mean = 0, sd = 1) +
  node("Y", distr = "rbern", prob = plogis(0.5 * M + 1.1 * R + 1.3 * A))
Dset <- set.DAG(D)
#> ...automatically assigning order attribute to some nodes...
#> node A, order:1
#> node P, order:2
#> node M, order:3
#> node R, order:4
#> node Y, order:5
```

Generate DAG

```r
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
#> using the following vertex attributes:
#> 120.8NAdarkbluenone0
#> using the following edge attributes:
#> 0.50.40.7black1
```

Generate Data

```r
require(simcausal)
Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
head(Obs.Data)
#>   ID           A P           M           R Y
#> 1  1 -0.56047565 0  1.17311294  0.49170637 1
#> 2  2 -0.23017749 0 -1.56302981 -1.17864299 1
#> 3  3  1.55870831 0  1.19759283  0.87393920 1
#> 4  4  0.07050839 0 -0.88415718 -0.09761158 1
#> 5  5  0.12928774 0  0.80065978  1.46814830 1
#> 6  6  1.71506499 0  0.02210438 -1.05044636 1
```

Estimate effect (OR)

```r
fit <- glm(Y ~ A + M, family=binomial(link = "logit"), data=Obs.Data)
round(coef(fit),2)
#> (Intercept)           A           M
#>        0.00        1.06        0.41

fit2 <- glm(Y ~ A + M + R, family=binomial(link = "logit"), data=Obs.Data)
round(coef(fit2),2)
#> (Intercept)           A           M           R
#>        0.00        1.29        0.50        1.10
```

Including a variable that is not a confounder (R is a pure risk factor for the outcome Y) in the model changes effect estimate (1.3).

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

R functions (R)

The list of new R functions introduced in this *Confounding and bias* lab component are below:

Function_name	Package_name	Use
cmh.test	lawstat	To conduct the Mantel-Haenszel Chi-square test
DAG.empty	simcausal	To initialize an empty DAG
ftable	base/stats	To create a flat contingency table
plotDAG	simcausal	To visualize a DAG
set.DAG	simcausal	To create a DAG
sim	simcausal	To simulate data using a DAG

Quiz (R)

> ⚠ Warning
>
> We are currently in the beta testing phase for our quiz page updates, so please be aware that some responses may not be accurate. Thank you for your patience.
> For those interested in downloading the quiz questions, you'll find a section below dedicated to downloading.

Live Quiz

You can click on the live quiz options to either see hints of why your response may be incorrect (response marked in red), or whether your choice of response is correct (response marked in green):

Download Quiz

Quiz (R)

1. **Downloading the File:**
    - Navigate to the link: See here.
    - Right-click on the link and select "Save link as…" from the dropdown menu.
    - Alternatively click here for downloading the quizzes.
    - Choose a destination folder on your computer where you'd like to save the file (e.g., Desktop). **Remember this location**, as you'll need to navigate to it later.

2. **Setting Up RStudio:**

   - If you don't have RStudio installed, see the download link in here.
   - Launch RStudio after installing.

3. **Installing Necessary R Packages:**

   - Before running the R Markdown file, ensure you have the required packages. In RStudio's console (located at the bottom left by default), enter the following commands to install them:

```
install.packages("learnr")
install.packages("xfun")
```

4. **Opening and Running the File:**

   - In RStudio, go to `File` > `Open File...`.
   - Navigate to the folder where you saved the `Rmd` file and select it to open.
   - Once the file is open in RStudio, you'll see a "Run Document" button (green) at the top of the script editor. Click on it to run the R Markdown Quiz file.

Part V

Prediction ideas

Background

The chapter provides a comprehensive guide to prediction modeling for cholesterol levels, focusing on the challenges and solutions involved in building robust prediction models. It begins by addressing the issue of collinearity among predictors and progresses to cover the intricacies of modeling both continuous and binary outcomes. Special attention is given to diagnosing and preventing model overfitting through various techniques such as data splitting and cross-validation. Advanced topics in model validation like bootstrapping are also explored. The overarching theme is to equip data analysts with the tools and methods needed to build, assess, and improve predictive models while addressing common challenges like collinearity and overfitting.

> **!** Important
>
> **Datasets**:
> All of the datasets used in this tutorial can be accessed from this GitHub repository folder

As we've journeyed through the previous chapters, we've gained a comprehensive understanding of various research questions, particularly distinguishing between causal and predictive inquiries. While the prior chapter delved into the intricacies of causal questions and the challenges they present, this chapter shifts the spotlight to the realm of prediction. Predictive questions have their own set of complexities and methodologies, distinct from those of causal inquiries. Here, we'll explore the art and science of making accurate predictions, understanding the factors that influence them, and the tools and techniques best suited for predictive analysis.

Furthermore, this chapter serves as a precursor to our upcoming exploration of machine learning. While prediction provides the foundation, machine learning offers advanced tools and algorithms to refine and enhance our predictive capabilities. By building on the foundational knowledge from the preceding chapters and setting the stage for the machine learning chapter, we aim to provide a holistic view of how prediction and machine learning intertwine in the broader landscape of research inquiry.

Overview of tutorials

Identify collinear predictors

This tutorial focuses on identifying collinear predictors in a dataset related to cholesterol levels from the NHANES 2015 collection. The tutorial guides you through summarizing its structure, and applying methods for variable clustering to detect collinear predictors. The tutorial is practical for data analysts aiming to improve model accuracy by identifying and addressing redundant variables.

Explore relationships for continuous outcome variable

This comprehensive tutorial walks you through the process of analyzing a dataset on cholesterol levels, focusing on exploring relationships for a continuous outcome variable. It starts by generating a correlation plot. Multiple methods for examining descriptive associations are provided, including stratification by key predictors. The tutorial also covers linear regression modeling, diagnosing data issues like outliers and leverage, and refitting the model after cleaning the data. Additionally, the tutorial delves into more complex modeling techniques like polynomial regression and multiple covariates, and addresses issues of collinearity using Variance Inflation Factors (VIF).

Explore relationships for binary outcome variable

A binary outcome variable is created to classify cholesterol levels as 'healthy' or 'unhealthy'. This transformed variable is then modeled using logistic regression. Various predictors including demographic variables, vital statistics, and other health parameters are considered in the model. The performance of the model is evaluated using Variance Inflation Factor (VIF) for multicollinearity and Area Under the Curve (AUC) for classification accuracy. Two models are fitted, and their respective AUCs are calculated to assess the predictive power.

Overfitting and performance

The tutorial focus is on addressing overfitting and assessing model performance. A linear regression model is fitted using a comprehensive set of predictors. Various statistical metrics such as the design matrix dimensions, Sum of Squares for Error (SSE), Total Sum of Squares (SST), R-squared (R2), Root Mean Square Error (RMSE), and Adjusted R2 are calculated to evaluate the model's predictive power and fit. Functions are also created to streamline the calculation of these metrics, allowing for more dynamic and customizable performance assessment. One such function, perform, encapsulates the entire process, outputting key performance indicators including R2,

adjusted R2, and RMSE, and it can be applied to new data sets for validation.

Data spliting

The tutorial focuses on splitting data into training and testing sets to prevent model overfitting. We allocate approximately 70% of the data to the training set and the remaining 30% to the test set. The linear regression model is then fitted using the training data. Performance metrics are extracted using the previously defined perform function, which is applied not only to the training and test sets but also to the entire dataset for comprehensive performance evaluation. This data splitting approach allows for more robust model validation by assessing how well the model generalizes to unseen data.

Cross-vaildation

The tutorial outlines the process of implementing k-fold cross-validation to validate a linear regression model's performance, aiming to predict cholesterol levels. The dataset is divided into 5 folds, by turn used as training (to fit the model), and test sets (used for prediction and performance evaluation). Performance metrics such as R-squared are calculated for each fold. The process can also be automated , which helps in fitting the model across all folds and summarizing the results, including calculating the mean and standard deviation of the R-squared values to understand the model's consistency and reliability.

Bootstrap

The tutorial outlines methods for implementing various bootstrapping techniques in statistical analysis. It demonstrates resampling methods using vectors and matrices. The idea of bootstrapping is emphasized as a useful technique for estimating the standard deviation (SD) of a statistic (e.g., mean), when the distribution of the data is unknown. This SD is then used to calculate confidence intervals. Different variations of bootstrap methods, such as "boot," "boot632," and "Optimism corrected

bootstrap," are demonstrated for linear regression and logistic regression models. They are used to obtain performance metrics like R-squared for regression models and the Receiver Operating Characteristic (ROC) curve for classification models. The tutorial also includes an example of calculating the Brier Score. The examples aim to offer various strategies for model evaluation, from the basics of resampling a vector to applying complex methods like 'Optimism corrected bootstrap' on real-world data.

> Tip
>
> **Optional Content**:
> You'll find that some sections conclude with an optional video walkthrough that demonstrates the code. Keep in mind that the content might have been updated since these videos were recorded. Watching these videos is optional.

> Warning
>
> **Bug Report**:
> Fill out this form to report any issues with the tutorial.

Concepts (P)

Research goals

Epidemiologists identify four types of inferential goals in research:

1. Prediction: Establishing models to predict future outcomes based on current or past information.
2. Evaluating an exposure of primary interest: Focusing on assessing the impact or significance of a specific exposure variable within a model.
3. Identifying the important independent predictors of an outcome: Determining which variables most significantly affect the outcome and understanding the strength and nature of these relationships.
4. Descriptive: A possible emphasis on describing the data and relationships within it.

Reading list

Key reference: (Vittinghoff et al. 2011a)

Optional reading:

- (Greenland and Pearce 2015)
- (Kuhn and Johnson 2013) (chapter 4)

Exercise:

Which type of goal does this article have?

- (Williamson et al. 2020)

Video Lessons

Inferential goals in an epidemiological study

Prediction, causal, important predictors, descriptive

Goal 1: Prediction model

discrimination, calibration, overfitting, validation, model selection

Goal 2: Causal exploration

outcome vs. exposure of primary interest

Goal 3: Outcome vs. multiple exposures

Centering and scaling

Choosing reference level

Criteria to determine the appropriate reference level for a categorical covariate:

- If the covariate possesses at least an ordinal nature and serves primarily as an adjustment variable, it is advisable to select either the lowest or highest category as the reference level. This choice can be particularly useful in uncovering potential dose-response relationships within the data.
- Consider the specific aspect you wish to emphasize in your interpretation. For instance, if you aim to shed light on the concept of an unhealthy diet within the context, opting for the "healthy" category as the reference level can align with the typical causal motivation behind the choice.

- In general, it is advisable to designate the category with the highest frequency as the reference level. This selection carries a statistical advantage, especially when dealing with imbalanced categories. Avoid choosing a low-frequency category as the reference level, as regression estimates may become highly unstable under such circumstances.

Video Lesson Slides

Links

- Google Slides
- PDF Slides

References

Collinear predictors

In this tutorial, we'll continue with the data analysis. We'll focus on an analysis of an NHANES data. This data contains over 1200 observations and 33 variables. These variables come in various types: numeric, categorical, and binary. Our primary goal is to fit a linear regression model to predict cholesterol levels.

```r
# Load required packages
library(rms)
library(Hmisc)
```

Load data

Let us load the dataset and see structure of the variables:

```r
load(file = "Data/predictivefactors/cholesterolNHANES15.RData")
#head(analytic)
str(analytic)
#> 'data.frame':    1267 obs. of  33 variables:
#>  $ ID                  : num  83732 83733 83741 83747 83750 ...
#>  $ gender              : chr  "Male" "Male" "Male" "Male" ...
#>  $ age                 : num  62 53 22 46 45 30 60 69 24 70 ...
#>  $ born                : chr  "Born in 50 US states or Washingt" "Others" "Born in 50 US s
#>  $ race                : chr  "White" "White" "Black" "White" ...
#>  $ education           : chr  "College" "High.School" "College" "College" ...
#>  $ married             : chr  "Married" "Previously.married" "Never.married" "Married" ...
#>  $ income              : chr  "Between.55kto99k" "<25k" "Between.25kto54k" "<25k" ...
#>  $ weight              : num  135630 25282 39353 35674 97002 ...
#>  $ psu                 : num  1 1 2 1 1 1 1 2 1 2 ...
#>  $ strata              : num  125 125 128 121 125 124 128 120 130 132 ...
#>  $ diastolicBP         : num  70 88 70 94 70 50 74 70 72 54 ...
#>  $ systolicBP          : num  128 146 110 144 116 104 142 146 126 144 ...
```

```
#>  $ bodyweight           : num  94.8 90.4 76.6 86.2 76.2 71.2 75.6 84 89.2 81.7 ...
#>  $ bodyheight           : num  184 171 165 177 178 ...
#>  $ bmi                  : num  27.8 30.8 28 27.6 24.1 26.6 35.9 31 26.9 27 ...
#>  $ waist                : num  101.1 107.9 86.6 104.3 90.1 ...
#>  $ smoke                : chr  "Not.at.all" "Every.day" "Some.days" "Every.day" ...
#>  $ alcohol              : num  1 6 8 1 3 2 1 1 2 2 ...
#>  $ cholesterol          : num  173 265 164 242 181 184 205 287 126 192 ...
#>  $ cholesterolM2        : num  4.47 6.85 4.24 6.26 4.68 4.76 5.3 7.42 3.26 4.97 ...
#>  $ triglycerides        : num  158 170 77 497 63 62 169 245 95 64 ...
#>  $ uric.acid            : num  4.2 7 6 6.5 5.4 5.5 5.1 4.3 7.6 7.1 ...
#>  $ protein              : num  7.5 7.4 7.4 6.8 7.4 6.7 7.4 6.8 7.3 7.2 ...
#>  $ bilirubin            : num  0.5 0.6 0.2 0.5 0.7 0.8 0.4 0.6 1.2 1.2 ...
#>  $ phosphorus           : num  4.7 4.4 5.3 3.6 3.9 3.4 3.9 4.4 3.2 3 ...
#>  $ sodium               : num  136 140 139 138 138 136 139 140 140 139 ...
#>  $ potassium            : num  4.3 4.55 4.16 4.27 3.91 3.97 3.99 4.25 3.8 4.63 ...
#>  $ globulin             : num  2.9 2.9 3 2.6 2.8 2.5 3.2 2.3 2.7 2.6 ...
#>  $ calcium              : num  9.8 9.8 9.3 9.3 9.3 9.4 9.6 9.6 9.6 9.6 ...
#>  $ physical.work        : chr  "No" "No" "No" "No" ...
#>  $ physical.recreational: chr  "No" "No" "Yes" "No" ...
#>  $ diabetes             : chr  "Yes" "No" "No" "No" ...
#>  - attr(*, "na.action")= 'omit' Named int [1:3739] 3 4 5 6 8 9 13 14 15 16 ...
#>   ..- attr(*, "names")= chr [1:3739] "3" "4" "5" "6" ...
```

Describe the data

```r
require(rms)
describe(analytic)
#> analytic
#>
#>  33  Variables      1267  Observations
#> ---------------------------------------------------------------------------
#> ID
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0     1267        1    88660     3366    84250    84687
#>      .25      .50      .75      .90      .95
#>    86019    88692    91252    92670    93089
#>
#> lowest : 83732 83733 83741 83747 83750, highest: 93617 93633 93643 93659 93685
```

```
#> ---------------------------------------------------------------------------------
#> gender
#>        n  missing distinct
#>     1267        0        2
#>
#> Value      Female   Male
#> Frequency     496    771
#> Proportion  0.391  0.609
#> ---------------------------------------------------------------------------------
#> age
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0       61        1    49.91    19.18       24       27
#>      .25      .50      .75      .90      .95
#>       36       51       63       72       78
#>
#> lowest : 20 21 22 23 24, highest: 76 77 78 79 80
#> ---------------------------------------------------------------------------------
#> born
#>        n  missing distinct
#>     1267        0        2
#>
#> Value      Born in 50 US states or Washingt                          Others
#> Frequency                                991                             276
#> Proportion                             0.782                           0.218
#> ---------------------------------------------------------------------------------
#> race
#>        n  missing distinct
#>     1267        0        4
#>
#> Value         Black Hispanic    Other    White
#> Frequency       246      337      132      552
#> Proportion    0.194    0.266    0.104    0.436
#> ---------------------------------------------------------------------------------
#> education
#>        n  missing distinct
#>     1267        0        3
#>
#> Value          College High.School      School
#> Frequency          648         523          96
#> Proportion       0.511       0.413       0.076
```

```
#> ------------------------------------------------------------------------------
#> married
#>        n  missing distinct
#>     1267        0        3
#>
#> Value                 Married      Never.married Previously.married
#> Frequency                 751                226                290
#> Proportion              0.593              0.178              0.229
#> ------------------------------------------------------------------------------
#> income
#>        n  missing distinct
#>     1267        0        4
#>
#> Value                  <25k Between.25kto54k Between.55kto99k         Over100k
#> Frequency               344              435              297              191
#> Proportion            0.272            0.343            0.234            0.151
#> ------------------------------------------------------------------------------
#> weight
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0     1184        1    48904    44337     9158    11549
#>      .25      .50      .75      .90      .95
#>    19540    30335    63822   121803   151546
#>
#> lowest :   5470.041   5948.955   6197.660   6480.947   6703.837
#> highest: 203562.855 207197.232 213611.345 218138.797 224891.623
#> ------------------------------------------------------------------------------
#> psu
#>        n  missing distinct     Info     Mean      Gmd
#>     1267        0        2     0.75    1.493   0.5003
#>
#> Value          1     2
#> Frequency    642   625
#> Proportion 0.507 0.493
#> ------------------------------------------------------------------------------
#> strata
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0       15    0.994    126.3    4.792      120      121
#>      .25      .50      .75      .90      .95
#>      123      126      130      132      133
#>
```

```
#> lowest : 119 120 121 122 123, highest: 129 130 131 132 133
#>
#> Value        119   120   121   122   123   124   125   126   127   128   129
#> Frequency     47    74   118    63    77    66   114   104   107    65    53
#> Proportion 0.037 0.058 0.093 0.050 0.061 0.052 0.090 0.082 0.084 0.051 0.042
#>
#> Value        130   131   132   133
#> Frequency     99   120    95    65
#> Proportion 0.078 0.095 0.075 0.051
#> ---------------------------------------------------------------------------
#> diastolicBP
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0       41    0.997    70.37    13.99       52       54
#>      .25      .50      .75      .90      .95
#>       62       70       78       86       92
#>
#> lowest :   0  26  34  38  40, highest: 104 106 108 110 112
#> ---------------------------------------------------------------------------
#> systolicBP
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0       56    0.998    126.5     19.3    102.0    106.0
#>      .25      .50      .75      .90      .95
#>    114.0    124.0    136.0    148.8    160.0
#>
#> lowest :  84  88  90  92  94, highest: 194 196 206 218 236
#> ---------------------------------------------------------------------------
#> bodyweight
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0      615        1    84.95    23.56    56.29    61.10
#>      .25      .50      .75      .90      .95
#>    69.70    81.40    97.00   113.44   127.47
#>
#> lowest :  39.7  39.8  40.7  42.6  42.7, highest: 161.9 166.3 175.7 175.9 178.4
#> ---------------------------------------------------------------------------
#> bodyheight
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0      376        1    169.2    10.66    153.8    157.0
#>      .25      .50      .75      .90      .95
#>    162.6    169.3    176.2    181.1    184.2
#>
```

```
#> lowest : 143.8 144.2 145.2 145.9 146.2, highest: 194.6 195.1 195.6 198.4 201.0
#> ---------------------------------------------------------------------------------
#> bmi
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0      284        1    29.58    7.403    20.60    22.06
#>      .25      .50      .75      .90      .95
#>    24.80    28.60    33.30    38.24    42.00
#>
#> lowest : 16.3 17.5 17.6 17.7 17.9, highest: 57.2 57.6 59.4 60.7 64.5
#> ---------------------------------------------------------------------------------
#> waist
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0      544        1    101.8    18.47     77.1     81.4
#>      .25      .50      .75      .90      .95
#>     90.5    100.3    111.2    122.8    132.5
#>
#> lowest :  65.0  65.5  66.5  68.2  68.7, highest: 159.2 159.8 160.2 160.5 161.5
#> ---------------------------------------------------------------------------------
#> smoke
#>        n  missing distinct
#>     1267        0        3
#>
#> Value       Every.day Not.at.all  Some.days
#> Frequency         448        665        154
#> Proportion      0.354      0.525      0.122
#> ---------------------------------------------------------------------------------
#> alcohol
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0       14    0.952    3.109    2.419        1        1
#>      .25      .50      .75      .90      .95
#>        1        2        4        6        8
#>
#> lowest :  1  2  3  4  5, highest: 10 11 12 14 15
#>
#> Value          1     2     3     4     5     6     7     8     9    10    11
#> Frequency    336   371   189   106    79    95    10    26     4    20     1
#> Proportion 0.265 0.293 0.149 0.084 0.062 0.075 0.008 0.021 0.003 0.016 0.001
#>
#> Value         12    14    15
#> Frequency     23     1     6
```

```
#> Proportion 0.018 0.001 0.005
#> ---------------------------------------------------------------------------------
#> cholesterol
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0      203        1    193.1    47.47    132.0    142.0
#>      .25      .50      .75      .90      .95
#>    162.5    191.0    217.0    248.0    268.0
#>
#> lowest :  81  93  97 100 101, highest: 345 348 349 358 545
#> ---------------------------------------------------------------------------------
#> cholesterolM2
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0      203        1    4.994    1.228    3.410    3.670
#>      .25      .50      .75      .90      .95
#>    4.205    4.940    5.610    6.410    6.930
#>
#> lowest :  2.09  2.40  2.51  2.59  2.61, highest:  8.92  9.00  9.03  9.26 14.09
#> ---------------------------------------------------------------------------------
#> triglycerides
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0      361        1    165.8    124.1     48.0     59.0
#>      .25      .50      .75      .90      .95
#>     84.0    127.0    201.5    309.0    396.6
#>
#> lowest :   18   21   24   25   31, highest:  964 1020 1157 1253 3061
#> ---------------------------------------------------------------------------------
#> uric.acid
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0       84        1    5.598    1.626     3.43     3.80
#>      .25      .50      .75      .90      .95
#>     4.60     5.50     6.50     7.40     8.00
#>
#> lowest :  1.6  2.2  2.3  2.4  2.5, highest: 10.2 10.3 11.7 12.2 18.0
#> ---------------------------------------------------------------------------------
#> protein
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0       32    0.995    7.126   0.5095      6.4      6.6
#>      .25      .50      .75      .90      .95
#>      6.8      7.1      7.4      7.7      7.9
#>
```

```
#> lowest : 5.7 5.8 5.9 6.0 6.1, highest: 8.4 8.5 8.6 8.8 9.0
#> ---------------------------------------------------------------------------
#> bilirubin
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0       31    0.984   0.5467   0.2949      0.2      0.2
#>      .25      .50      .75      .90      .95
#>      0.4      0.5      0.7      0.9      1.0
#>
#> lowest : 0.00 0.01 0.02 0.03 0.04, highest: 1.80 2.00 2.10 2.60 3.30
#> ---------------------------------------------------------------------------
#> phosphorus
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0       37    0.996    3.642    0.593      2.8      3.0
#>      .25      .50      .75      .90      .95
#>      3.3      3.6      4.0      4.3      4.5
#>
#> lowest : 1.8 2.0 2.2 2.3 2.4, highest: 5.2 5.3 5.4 5.6 6.1
#> ---------------------------------------------------------------------------
#> sodium
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0       20    0.977    138.5    2.383      135      136
#>      .25      .50      .75      .90      .95
#>      137      139      140      141      142
#>
#> lowest : 124 126 129 130 131, highest: 142 143 144 146 148
#>
#> Value        124   126   129   130   131   132   133   134   135   136   137
#> Frequency      1     1     1     1     5     4    11    23    46    93   176
#> Proportion 0.001 0.001 0.001 0.001 0.004 0.003 0.009 0.018 0.036 0.073 0.139
#>
#> Value        138   139   140   141   142   143   144   146   148
#> Frequency    235   260   206   112    55    29     6     1     1
#> Proportion 0.185 0.205 0.163 0.088 0.043 0.023 0.005 0.001 0.001
#> ---------------------------------------------------------------------------
#> potassium
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0      175        1    3.985   0.3725     3.45     3.57
#>      .25      .50      .75      .90      .95
#>     3.78     3.98     4.19     4.40     4.54
#>
```

```
#> lowest : 2.60 2.92 2.96 3.07 3.09, highest: 5.15 5.21 5.36 5.37 5.51
#> ---------------------------------------------------------------------------
#> globulin
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0       29    0.994    2.799   0.4536      2.2      2.3
#>      .25      .50      .75      .90      .95
#>      2.5      2.8      3.0      3.3      3.5
#>
#> lowest : 1.6 1.8 1.9 2.0 2.1, highest: 4.1 4.2 4.3 4.5 5.5
#> ---------------------------------------------------------------------------
#> calcium
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     1267        0       25    0.991    9.335   0.3786      8.8      8.9
#>      .25      .50      .75      .90      .95
#>      9.1      9.3      9.6      9.7      9.9
#>
#> lowest :  8.4  8.5  8.6  8.7  8.8, highest: 10.4 10.5 10.7 11.0 11.1
#> ---------------------------------------------------------------------------
#> physical.work
#>        n  missing distinct
#>     1267        0        2
#>
#> Value         No   Yes
#> Frequency    895   372
#> Proportion 0.706 0.294
#> ---------------------------------------------------------------------------
#> physical.recreational
#>        n  missing distinct
#>     1267        0        2
#>
#> Value         No   Yes
#> Frequency   1002   265
#> Proportion 0.791 0.209
#> ---------------------------------------------------------------------------
#> diabetes
#>        n  missing distinct
#>     1267        0        2
#>
#> Value        No  Yes
#> Frequency  1064  203
```

```
#> Proportion 0.84 0.16
#> ----------------------------------------------------------------------------------------
```

Collinearity

Avoiding collinear variables can result in a more interpretable, stable, and efficient predictive model. Collinearity refers to a situation in which two or more predictor variables in a multiple regression model are highly correlated, meaning that one can be linearly predicted from the others with substantial accuracy. Collinearity poses several issues for predictive analysis:

Reduced Interpretability: When predictor variables are highly correlated, it becomes challenging to isolate the impact of individual predictors on the response variable. In other words, it is difficult to determine which predictor is genuinely influential in explaining variance in the response variable. This reduces the interpretability of the model.

Unstable Coefficients: Collinearity can lead to inflated standard errors of the regression coefficients. This means that the coefficients can be very sensitive to small changes in the data, making the model unstable and less generalizable to new, unseen data.

Overfitting: When predictors are collinear, the model is more likely to fit to the noise in the data rather than the actual signal. This is a manifestation of overfitting, where the model becomes too complex and captures random noise. Overfitted models will perform poorly on new, unseen data.

Inefficiency: Including redundant variables (collinear variables) does not add additional information to the model. This could be inefficient, especially when dealing with large datasets, as it increases computational costs without improving model performance.

Multicollinearity Diagnostics

Several techniques are available for diagnosing multicollinearity, including:

- Variance Inflation Factor (VIF): The VIF is a measure of multicollinearity, which measures how much the variance of a regression coefficient is increased because of multicollinearity. A general rule of thumb is that if VIF exceeds 4 or 5, multicollinearity is present.
- Eigenvalues and Eigenvectors of the correlation/covariance matrix: The eigenvalues of the correlation matrix can also be used to measure the presence of multicollinearity, with one or more eigenvalues close to zero indicating multicollinearity.
- Pairwise correlation matrices: Large correlation coefficients in the correlation matrix of predictors indicate the possibility of multicollinearity.

Remedies

Some common ways to handle collinearity include:

- Removing one of the correlated predictors
- Combining correlated predictors into a single composite predictor
- Using regularization techniques like Ridge Regression, which can help handle collinearity by adding a penalty term. Ridge regression adds a small amount of bias to the regression estimates, which in turn reduces the standard error, addresses the collinearity issue to some extent, and makes the results more reliable.

Identify collinear predictors

> **Tip**
>
> We can also use `hclust` and `varclus` or variable clustering, i.e., to identify collinear predictors

`hclust` is the hierarchical clustering function where default is squared Spearman correlation coefficients to detect monotonic but nonlinear relationships.

```r
require(Hmisc)
sel.names <- c("gender", "age", "born", "race", "education",
               "married", "income", "diastolicBP", "systolicBP",
               "bodyweight", "bodyheight", "bmi", "waist", "smoke",
               "alcohol",  "cholesterol", "triglycerides",
               "uric.acid", "protein", "bilirubin", "phosphorus",
               "sodium", "potassium", "globulin", "calcium",
               "physical.work", "physical.recreational",
               "diabetes")
var.cluster <- varclus(~., data = analytic[sel.names])
# var.cluster
plot(var.cluster)
```

In this plot, Spearman correlation or Spearman's ρ(rho) assesses the monotonic but nonlinear relationships predictors. We can also specify linear relationships such as Pearson correlation. A high correlation coefficient indicates predictors are highly correlated. For example, Spearman's ρ is more than 0.8 for bodyweight, BMI, and waist, indicating high collinearity between these predictors.

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Continuous outcome

Let us focus on issues related to predictive modeling for continuous outcomes in 4 steps:

- Diagnosis Phase: Identifies outliers, leverage points, and residuals that could affect the model.
- Cleaning Phase: Deletes problematic data based on predefined conditions.
- Modeling Phase: Various models are fitted including polynomial models and a comprehensive model with multiple predictors.
- Colinearity Check: A rule of thumb is used to check for multicollinearity in the comprehensive model, and problematic variables are flagged.

Explore relationships for continuous outcome variable

First, load several R packages for statistical modeling, data manipulation, and visualization.

```r
# Load required packages
library(rms)
library(Hmisc)
library(dplyr)
library(Publish)
library(car)
library(corrplot)
library(olsrr)
```

Load data

Here, a dataset is loaded into the R environment from an RData file.

```
load(file = "Data/predictivefactors/cholesterolNHANES15.RData")
```

Correlation plot

> **Tip**
>
> We can use the `cor` function to see the correlation between numeric variables and then use the `corrplot` function to plot the `cor` object. The plot helps in understanding the linear or nonlinear relationships between different numerical variables.

```
require(corrplot)
numeric.names <- c("age", "diastolicBP", "systolicBP", "bodyweight",
                   "bodyheight", "bmi", "waist", "alcohol",
                   "cholesterol", "triglycerides", "uric.acid",
                   "protein", "bilirubin", "phosphorus", "sodium",
                   "potassium", "globulin", "calcium")
correlationMatrix <- cor(analytic[numeric.names])
mat.num <- round(correlationMatrix,2)
mat.num[mat.num>0.8 & mat.num < 1]
#> [1] 0.89 0.90 0.89 0.91 0.90 0.91
corrplot(correlationMatrix, method="number", type="upper")
```

As we can see, the correlation between bmi and waist is 0.91, suggesting a high correlation between these two predictors. Similarly, bodyweight is highly correlated with waist (correlation 0.90), bodyweight is highly correlated with bmi (correlation 0.89). Also, there is no linear relationship between age and bmi (correlation -0.02).

Examine descriptive associations

Let us examine the descriptive associations with the dependent variable by stratifying separately by key predictors

> **Tip**
>
> There are multiple ways to examine the descriptive associations by strata/groups, e.g., `summarize`, `aggregate`, `describeBy`, `tapply`, `summary`

The code calculates and explores various ways to describe the cholesterol levels, stratified by key predictors such as gender.

```
mean(analytic$cholesterol)
#> [1] 193.1002

# Process 1
```

```r
mean(analytic$cholesterol[analytic$gender == "Male"])
#> [1] 190.7626
mean(analytic$cholesterol[analytic$gender == "Female"])
#> [1] 196.7339

# Process 2
library(dplyr)
analytic %>%
  group_by(gender) %>%
  dplyr::summarize(mean.ch=mean(cholesterol), .groups = 'drop')
#> # A tibble: 2 x 2
#>   gender mean.ch
#>   <chr>    <dbl>
#> 1 Female    197.
#> 2 Male      191.

# process 3
with(analytic, aggregate( analytic$cholesterol, by=list(gender),
                          FUN=summary))
#>   Group.1   x.Min. x.1st Qu. x.Median   x.Mean x.3rd Qu.   x.Max.
#> 1  Female 100.0000  166.7500 194.5000 196.7339  220.2500 358.0000
#> 2    Male  81.0000  161.0000 188.0000 190.7626  215.0000 545.0000

# process 4
psych::describeBy(analytic$cholesterol, analytic$gender)
#>
#>  Descriptive statistics by group
#> group: Female
#>    vars   n   mean    sd median trimmed   mad min max range skew kurtosis   se
#> X1    1 496 196.73 43.26  194.5  194.44 40.77 100 358   258 0.57     0.56 1.94
#> ------------------------------------------------------------
#> group: Male
#>    vars   n   mean    sd median trimmed   mad min max range skew kurtosis   se
#> X1    1 771 190.76 43.06    188  188.54 40.03  81 545   464  1.1     5.76 1.55

# process 5
tapply(analytic$cholesterol, analytic$gender, summary)
#> $Female
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   100.0   166.8   194.5   196.7   220.2   358.0
```

```
#>
#> $Male
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    81.0   161.0   188.0   190.8   215.0   545.0

# A general process
sel.names <- c("gender", "age", "born", "race", "education",
               "married", "income", "diastolicBP", "systolicBP",
               "bodyweight", "bodyheight", "bmi", "waist",
               "smoke", "alcohol", "cholesterol",
               "triglycerides", "uric.acid", "protein",
               "bilirubin", "phosphorus", "sodium", "potassium",
               "globulin", "calcium", "physical.work",
               "physical.recreational", "diabetes")
var.summ <- summary(cholesterol~ ., data = analytic[sel.names])
var.summ
#> cholesterol      N= 1267
#>
#> +--------------------+-------------------------------+----+-----------+
#> |                    |                               |   N|cholesterol|
#> +--------------------+-------------------------------+----+-----------+
#> |              gender|                         Female| 496|   196.7339|
#> |                    |                           Male| 771|   190.7626|
#> +--------------------+-------------------------------+----+-----------+
#> |                 age|                       [20,37)| 342|   182.4854|
#> |                    |                       [37,52)| 313|   200.1661|
#> |                    |                       [52,64)| 315|   199.7873|
#> |                    |                       [64,80]| 297|   190.7845|
#> +--------------------+-------------------------------+----+-----------+
#> |                born|Born in 50 US states or Washingt| 991|   190.9253|
#> |                    |                         Others| 276|   200.9094|
#> +--------------------+-------------------------------+----+-----------+
#> |                race|                          Black| 246|   187.3740|
#> |                    |                       Hispanic| 337|   193.5490|
#> |                    |                          Other| 132|   191.8561|
#> |                    |                          White| 552|   195.6757|
#> +--------------------+-------------------------------+----+-----------+
#> |           education|                        College| 648|   192.5478|
#> |                    |                    High.School| 523|   193.4532|
#> |                    |                         School|  96|   194.9062|
```

```
#> +--------------------+-------------------------------+----+-----------+
#> |             married|                        Married| 751|   194.0306|
#> |                    |                  Never.married| 226|   182.8761|
#> |                    |             Previously.married| 290|   198.6586|
#> +--------------------+-------------------------------+----+-----------+
#> |              income|                           <25k| 344|   191.9564|
#> |                    |               Between.25kto54k| 435|   191.9310|
#> |                    |               Between.55kto99k| 297|   195.7508|
#> |                    |                       Over100k| 191|   193.7016|
#> +--------------------+-------------------------------+----+-----------+
#> |          diastolicBP|                       [ 0, 64)| 336|   186.7649|
#> |                    |                       [64, 72)| 321|   189.3458|
#> |                    |                       [72, 80)| 319|   195.7085|
#> |                    |                       [80,112]| 291|   201.6976|
#> +--------------------+-------------------------------+----+-----------+
#> |           systolicBP|                      [ 84,116)| 340|   186.2765|
#> |                    |                      [116,126)| 317|   190.6372|
#> |                    |                      [126,138)| 335|   196.9881|
#> |                    |                      [138,236]| 275|   199.6400|
#> +--------------------+-------------------------------+----+-----------+
#> |           bodyweight|                    [39.7, 69.8)| 319|   193.8903|
#> |                    |                    [69.8, 81.5)| 316|   197.1424|
#> |                    |                    [81.5, 97.2)| 317|   192.4984|
#> |                    |                    [97.2,178.4]| 315|   188.8508|
#> +--------------------+-------------------------------+----+-----------+
#> |           bodyheight|                     [144,163)| 317|   198.7003|
#> |                    |                     [163,169)| 320|   193.7750|
#> |                    |                     [169,176)| 314|   189.8790|
#> |                    |                     [176,201]| 316|   190.0000|
#> +--------------------+-------------------------------+----+-----------+
#> |                 bmi|                    [16.3,24.9)| 322|   188.8043|
#> |                    |                    [24.9,28.7)| 315|   198.5016|
#> |                    |                    [28.7,33.4)| 317|   197.5016|
#> |                    |                    [33.4,64.5]| 313|   187.6262|
#> +--------------------+-------------------------------+----+-----------+
#> |               waist|                  [ 65.0, 90.7)| 320|   188.9688|
#> |                    |                  [ 90.7,100.4)| 315|   199.6413|
#> |                    |                  [100.4,111.3)| 316|   197.3892|
#> |                    |                  [111.3,161.5]| 316|   186.4747|
#> +--------------------+-------------------------------+----+-----------+
```

```
#> |                smoke|                       Every.day| 448|   191.5938|
#> |                     |                     Not.at.all| 665|   194.6451|
#> |                     |                      Some.days| 154|   190.8117|
#> +---------------------+--------------------------------+----+-----------+
#> |              alcohol|                              1| 336|   191.0387|
#> |                     |                              2| 371|   192.0809|
#> |                     |                         [3, 5)| 295|   195.9356|
#> |                     |                        [5,15]| 265|   193.9849|
#> +---------------------+--------------------------------+----+-----------+
#> |        triglycerides|                     [ 18,  85)| 320|   172.2344|
#> |                     |                     [ 85, 128)| 319|   185.6834|
#> |                     |                    [128, 203)| 314|   199.4140|
#> |                     |                   [203,3061]| 314|   215.5860|
#> +---------------------+--------------------------------+----+-----------+
#> |            uric.acid|                    [1.6, 4.7)| 348|   188.9310|
#> |                     |                    [4.7, 5.6)| 305|   191.8033|
#> |                     |                    [5.6, 6.6)| 307|   195.7720|
#> |                     |                   [6.6,18.0]| 307|   196.4430|
#> +---------------------+--------------------------------+----+-----------+
#> |              protein|                     [5.7,6.9)| 336|   189.8631|
#> |                     |                     [6.9,7.2)| 328|   192.3201|
#> |                     |                     [7.2,7.5)| 310|   193.4258|
#> |                     |                     [7.5,9.0]| 293|   197.3413|
#> +---------------------+--------------------------------+----+-----------+
#> |            bilirubin|                     [0.0,0.5)| 506|   195.7391|
#> |                     |                           0.5| 212|   192.2264|
#> |                     |                     [0.6,0.8)| 310|   192.0645|
#> |                     |                     [0.8,3.3]| 239|   189.6318|
#> +---------------------+--------------------------------+----+-----------+
#> |           phosphorus|                     [1.8,3.4)| 362|   188.0387|
#> |                     |                     [3.4,3.7)| 309|   192.5405|
#> |                     |                     [3.7,4.1)| 323|   195.5542|
#> |                     |                     [4.1,6.1]| 273|   197.5421|
#> +---------------------+--------------------------------+----+-----------+
#> |               sodium|                     [124,138)| 362|   191.9420|
#> |                     |                     [138,140)| 495|   194.2929|
#> |                     |                           140| 206|   191.7864|
#> |                     |                     [141,148]| 204|   193.5882|
#> +---------------------+--------------------------------+----+-----------+
#> |            potassium|                   [2.60,3.79)| 320|   191.5375|
```

```
#> |                     |                     [3.79,3.99)| 328|   192.3628|
#> |                     |                     [3.99,4.20)| 308|   196.9643|
#> |                     |                     [4.20,5.51]| 311|   191.6592|
#> +---------------------+--------------------------------+----+-----------+
#> |             globulin|                       [1.6,2.6)| 350|   189.9429|
#> |                     |                       [2.6,2.9)| 388|   199.0052|
#> |                     |                       [2.9,3.1)| 230|   193.3783|
#> |                     |                       [3.1,5.5]| 299|   188.9197|
#> +---------------------+--------------------------------+----+-----------+
#> |              calcium|                      [8.4, 9.2)| 371|   186.0323|
#> |                     |                      [9.2, 9.4)| 294|   188.3605|
#> |                     |                      [9.4, 9.7)| 395|   197.4430|
#> |                     |                      [9.7,11.1]| 207|   204.2126|
#> +---------------------+--------------------------------+----+-----------+
#> |        physical.work|                             No| 895|   194.0078|
#> |                     |                            Yes| 372|   190.9167|
#> +---------------------+--------------------------------+----+-----------+
#> |physical.recreational|                             No|1002|   193.5359|
#> |                     |                            Yes| 265|   191.4528|
#> +---------------------+--------------------------------+----+-----------+
#> |             diabetes|                             No|1064|   194.8036|
#> |                     |                            Yes| 203|   184.1724|
#> +---------------------+--------------------------------+----+-----------+
#> |              Overall|                               |1267|   193.1002|
#> +---------------------+--------------------------------+----+-----------+
plot(var.summ)
```

```r

summary(analytic$diastolicBP)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    0.00   62.00   70.00   70.37   78.00  112.00

analytic$diastolicBP[analytic$diastolicBP == 0] <- NA

# Bivariate Summaries Computed Separately by a Series of Predictors
var.summ2 <- spearman2(cholesterol~ ., data = analytic[sel.names])
plot(var.summ2)
```

Regression: Linear regression

A linear regression model is fitted to explore the association between cholesterol levels and triglycerides. Various summary statistics are also generated for the model.

> **Tip**
>
> We use `lm` function to fit the linear regression

```r
# set up formula with just 1 variable
formula0 <- as.formula("cholesterol~triglycerides")

# fitting regression on the analytic2 data
fit0 <- lm(formula0,data = analytic2)

# extract results
summary(fit0)
#>
#> Call:
#> lm(formula = formula0, data = analytic2)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -111.651  -26.157   -2.661   22.549  166.752
#>
#> Coefficients:
#>                Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   1.716e+02  1.127e+00  152.23   <2e-16 ***
#> triglycerides 1.275e-01  5.456e-03   23.37   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 37.38 on 2632 degrees of freedom
#> Multiple R-squared:  0.1718, Adjusted R-squared:  0.1715
#> F-statistic:   546 on 1 and 2632 DF,  p-value: < 2.2e-16

# extract just the coefficients/estimates
coef(fit0)
#>   (Intercept) triglycerides
```

```r
#>   171.6147531      0.1274909

# extract confidence intervals
confint(fit0)
#>                     2.5 %      97.5 %
#> (Intercept)   169.4042284 173.8252779
#> triglycerides   0.1167919   0.1381899

# residual plots
layout(matrix(1:6, byrow = T, ncol = 3))
plot(fit0, which = 1:6)
```

Diagnosis

0.0.0.0.1 * Identifying problematic data

Outliers: We can begin by plotting cholesterol against triglycerides to visualize any potential outliers. We can then identify data points where triglycerides are high.

Leverage: It calculates and plots leverage points. Leverage points that have values greater than 0.05 are isolated for inspection.

Residuals: Studentized residuals are computed for each data point to identify potential outliers. Those with values less than -5 are identified.

```r
require(olsrr)
# Outlier
plot(cholesterol ~ triglycerides, data = analytic2)
```

```r
subset(analytic2, triglycerides > 1500)
#>         ID gender age                            born  race education married
#> 4287 88018   Male  48 Born in 50 US states or Washingt White   College Married
#> 4845 88576   Male  33 Born in 50 US states or Washingt Black   College Married
#>                income    weight psu strata diastolicBP systolicBP bodyweight
#> 4287         Over100k 166919.97   2    121          78        128       90.4
#> 4845 Between.55kto99k  25026.18   2    129          64        120      103.6
#>      bodyheight  bmi waist alcohol cholesterol cholesterolM2 triglycerides
#> 4287      181.1 27.6  96.5       2         545         14.09          3061
#> 4845      179.9 32.0 108.0       1         433         11.20          1605
#>      uric.acid protein bilirubin phosphorus sodium potassium globulin calcium
#> 4287       5.3     6.7       0.5        3.1    134      4.16      2.5     9.1
#> 4845       6.5     7.7       0.8        3.5    133      3.69      3.1    10.2
#>      physical.work physical.recreational diabetes
#> 4287           Yes                    Yes       No
#> 4845            No                    Yes      Yes
```

```r
# leverage
ols_plot_resid_lev(fit0)
```

```r
analytic2$lev <- hat(model.matrix(fit0))
plot(analytic2$lev)
```

```r
summary(analytic2$lev)
#>      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
```

```r
#> 0.0003796 0.0004062 0.0004773 0.0007593 0.0005831 0.1800021
which(analytic2$lev > 0.05)
#> [1] 1102
subset(analytic2, lev > 0.05)
#>         ID gender age                              born  race education married
#> 4287 88018   Male  48 Born in 50 US states or Washingt White   College Married
#>        income weight psu strata diastolicBP systolicBP bodyweight bodyheight
#> 4287 Over100k 166920   2    121          78        128       90.4      181.1
#>       bmi waist alcohol cholesterol cholesterolM2 triglycerides uric.acid
#> 4287 27.6  96.5       2         545         14.09          3061       5.3
#>      protein bilirubin phosphorus sodium potassium globulin calcium
#> 4287     6.7       0.5        3.1    134      4.16      2.5     9.1
#>      physical.work physical.recreational diabetes       lev
#> 4287           Yes                    Yes       No 0.1800021

# Residual
analytic2$rstudent.values <- rstudent(fit0)
plot(analytic2$rstudent.values)
```

```r
which(analytic2$rstudent.values < -5)
#> integer(0)
# Heteroskedasticity: Test for constant variance
#ols_test_breusch_pagan(fit0, rhs = TRUE)
```

0.0.0.0.2 * Deleting suspicious data

We then delete observations based on two conditions: triglycerides > 1500 and leverage > 0.05.

```
# condition 1: triglycerides above 1500 needs deleting
analytic2b <- subset(analytic2, triglycerides < 1500)
dim(analytic2b)
#> [1] 2632   34

# condition 2: leverage above 0.05 needs deleting
analytic3 <- subset(analytic2b, lev < 0.05)
dim(analytic3)
#> [1] 2632   34

# Check how many observations are deleted
nrow(analytic2)-nrow(analytic3)
#> [1] 2
```

0.0.0.0.3 * Refitting in cleaned data

We refit the linear model on this cleaned data, and diagnostic plots are generated.

```
### Re-fit in data analytic3 (without problematic data)
formula0
#> cholesterol ~ triglycerides
fit0 <- lm(formula0,data = analytic3)

require(Publish)
publish(fit0)
#>       Variable Units Coefficient           CI.95 p-value
#>    (Intercept)            171.74 [169.37;174.11] < 1e-04
#>  triglycerides              0.13     [0.11;0.14] < 1e-04
layout(matrix(1:6, byrow = T, ncol = 3))
plot(fit0, which = 1:6)
```

```
require(car)
# component+residual plot or partial-residual plot
crPlots(fit0)
```

0.0.0.0.4 * Polynomial order 2

We fit polynomial models of orders 2 and 3 to explore non-linear relationships between cholesterol and triglycerides.

```r
formula1 <- as.formula("cholesterol~poly(triglycerides,2)")
formula1 <- as.formula("cholesterol~triglycerides^2")
fit1 <- lm(formula1,data = analytic3)
publish(fit1)
#>       Variable Units Coefficient           CI.95 p-value
#>    (Intercept)            171.74 [169.37;174.11] < 1e-04
#>  triglycerides              0.13     [0.11;0.14] < 1e-04

# Partial Residual Plots
crPlots(fit1)
```

It seems triglyceride has a quadratic effect on cholesterol. We can compare models fit0 and fit1 using the `anova` function:

```r
# compare fit0 and fit1 models
res0 <- anova(fit0,fit1)
print(res0)
#> Analysis of Variance Table
#>
#> Model 1: cholesterol ~ triglycerides
#> Model 2: cholesterol ~ triglycerides^2
#>   Res.Df     RSS Df Sum of Sq F Pr(>F)
#> 1   2630 3673461
#> 2   2630 3673461  0         0
```

0.0.0.0.5 * Polynomial order 3

```
# Fit a polynomial of order 3
formula2 <- as.formula("cholesterol~poly(triglycerides,3)")
formula2 <- as.formula("cholesterol~triglycerides^3")
fit2 <- lm(formula2,data = analytic3)
publish(fit2)
#>       Variable Units Coefficient           CI.95 p-value
#>    (Intercept)            171.74 [169.37;174.11] < 1e-04
#>  triglycerides              0.13     [0.11;0.14] < 1e-04

# Partial Residual Plots
crPlots(fit2)
```

```
# compare fit1 and fit2 models
res1 <- anova(fit1,fit2)
print(res1)
#> Analysis of Variance Table
#>
#> Model 1: cholesterol ~ triglycerides^2
#> Model 2: cholesterol ~ triglycerides^3
#>   Res.Df     RSS Df Sum of Sq F Pr(>F)
#> 1   2630 3673461
#> 2   2630 3673461  0         0
```

0.0.0.0.6 * Multiple covariates

We add more covariates.

```
# include everything!
formula3 <- as.formula("cholesterol ~ gender + age + born + race +
                       education + married + income + diastolicBP +
                       systolicBP + bmi + bodyweight + bodyheight +
                       waist +  triglycerides + uric.acid + protein +
                       bilirubin + phosphorus + sodium + potassium +
                       globulin + calcium + physical.work +
                       physical.recreational + diabetes")
fit3 <- lm(formula3, data = analytic3)
publish(fit3)
#>               Variable                          Units Coefficient           CI.95     p-va
#>            (Intercept)                                     280.02 [133.42;426.62]   0.000
#>                 gender                         Female         Ref
#>                                                  Male      -11.99  [-16.41;-7.57]     < 1
#>                    age                                      0.35     [0.23;0.47]     < 1
#>                   born Born in 50 US states or Washingt         Ref
#>                                                 Others        7.52    [3.68;11.36]   0.000
#>                   race                          Black         Ref
#>                                                Hispanic       -6.15  [-10.87;-1.44]   0.010
#>                                                   Other       -5.37   [-10.92;0.18]   0.057
#>                                                   White       -0.95    [-5.21;3.30]   0.660
#>              education                        College         Ref
#>                                             High.School        2.90    [-0.28;6.08]   0.074
#>                                                  School       -2.54    [-8.61;3.54]   0.413
#>                married                        Married         Ref
#>                                           Never.married       -5.72   [-9.63;-1.81]   0.004
#>                                       Previously.married        0.31    [-3.54;4.17]   0.873
#>                 income                           <25k         Ref
#>                                         Between.25kto54k       -0.97    [-4.87;2.93]   0.626
#>                                         Between.55kto99k        2.29    [-1.98;6.56]   0.292
#>                                                 Over100k        2.44    [-2.27;7.14]   0.309
#>            diastolicBP                                          0.38     [0.25;0.50]     < 1
#>             systolicBP                                          0.02    [-0.08;0.12]   0.666
#>                    bmi                                         -2.55   [-4.29;-0.81]   0.004
#>             bodyweight                                          0.82     [0.19;1.45]   0.010
#>             bodyheight                                         -0.89   [-1.55;-0.24]   0.007
#>                  waist                                         -0.02    [-0.29;0.26]   0.902
```

```
#>          triglycerides                                  0.12     [0.11;0.14]     < 1
#>              uric.acid                                  1.27     [0.08;2.47]   0.036
#>                protein                                  4.99   [-0.77;10.74]   0.089
#>              bilirubin                                 -5.43  [-10.53;-0.33]   0.037
#>             phosphorus                                 -0.18    [-2.81;2.45]   0.893
#>                 sodium                                 -0.97   [-1.66;-0.29]   0.005
#>              potassium                                  1.04    [-3.44;5.52]   0.648
#>               globulin                                 -2.25    [-8.22;3.71]   0.459
#>                calcium                                 12.02    [6.98;17.07]     < 1
#>          physical.work                          No       Ref
#>                                                 Yes     -0.45    [-3.68;2.79]   0.785
#>  physical.recreational                          No       Ref
#>                                                 Yes      1.35    [-1.94;4.65]   0.421
#>                diabetes                         No       Ref
#>                                                 Yes    -19.11 [-23.37;-14.85]     < 1
```

0.0.0.0.7 * Colinearity

We finally check for multicollinearity among predictors using the Variance Inflation Factor (VIF).

> **Note**
>
> Rule of thumb: variables with VIF > 4 needs further investigation

```
car::vif(fit3)
#>                          GVIF Df GVIF^(1/(2*Df))
#> gender                2.694171  1        1.641393
#> age                   2.164388  1        1.471186
#> born                  1.611478  1        1.269440
#> race                  2.463445  3        1.162137
#> education             1.435876  2        1.094660
#> married               1.481141  2        1.103187
#> income                1.402249  3        1.057964
#> diastolicBP           1.271126  1        1.127442
#> systolicBP            1.594986  1        1.262928
#> bmi                  81.811969  1        9.044997
#> bodyweight          101.102349  1       10.054966
#> bodyheight           21.863188  1        4.675809
```

```
#> waist                  11.913719  1        3.451626
#> triglycerides           1.219331  1        1.104233
#> uric.acid               1.603290  1        1.266211
#> protein                 3.622385  1        1.903256
#> bilirubin               1.185035  1        1.088593
#> phosphorus              1.116982  1        1.056874
#> sodium                  1.120920  1        1.058735
#> potassium               1.178381  1        1.085533
#> globulin                3.371211  1        1.836086
#> calcium                 1.591677  1        1.261617
#> physical.work           1.087315  1        1.042744
#> physical.recreational   1.226830  1        1.107624
#> diabetes                1.210715  1        1.100325
collinearity <- ols_vif_tol(fit3)
collinearity
#>                    Variables   Tolerance        VIF
#> 1                 genderMale 0.371171648   2.694171
#> 2                        age 0.462024352   2.164388
#> 3                 bornOthers 0.620548521   1.611478
#> 4               raceHispanic 0.398271889   2.510848
#> 5                  raceOther 0.514996433   1.941761
#> 6                  raceWhite 0.418431280   2.389879
#> 7       educationHigh.School 0.818384646   1.221919
#> 8            educationSchool 0.752526988   1.328856
#> 9       marriedNever.married 0.760566978   1.314809
#> 10 marriedPreviously.married 0.805388372   1.241637
#> 11    incomeBetween.25kto54k 0.547265531   1.827267
#> 12    incomeBetween.55kto99k 0.528356768   1.892661
#> 13            incomeOver100k 0.485459308   2.059905
#> 14               diastolicBP 0.786704109   1.271126
#> 15                systolicBP 0.626964572   1.594986
#> 16                       bmi 0.012223150  81.811969
#> 17                bodyweight 0.009890967 101.102349
#> 18                bodyheight 0.045738984  21.863188
#> 19                     waist 0.083936847  11.913719
#> 20             triglycerides 0.820121824   1.219331
#> 21                 uric.acid 0.623717352   1.603290
#> 22                   protein 0.276061222   3.622385
#> 23                 bilirubin 0.843856689   1.185035
#> 24                phosphorus 0.895269366   1.116982
```

```
#> 25                    sodium 0.892124045   1.120920
#> 26                 potassium 0.848621797   1.178381
#> 27                  globulin 0.296629287   3.371211
#> 28                   calcium 0.628268255   1.591677
#> 29          physical.workYes 0.919696951   1.087315
#> 30  physical.recreationalYes 0.815108557   1.226830
#> 31               diabetesYes 0.825958210   1.210715

# VIF > 4
collinearity[collinearity$VIF>4,]
#>     Variables   Tolerance       VIF
#> 16        bmi 0.012223150  81.81197
#> 17 bodyweight 0.009890967 101.10235
#> 18 bodyheight 0.045738984  21.86319
#> 19      waist 0.083936847  11.91372
```

```
formula4 <- as.formula("cholesterol ~ gender + age + born +
             race + education + married +
             income + diastolicBP + systolicBP +
             bmi + # bodyweight + bodyheight + waist +
             triglycerides + uric.acid +
             protein + bilirubin + phosphorus + sodium + potassium +
             globulin + calcium + physical.work + physical.recreational +
             diabetes")
fit4 <- lm(formula4, data = analytic3)
publish(fit4)
#>               Variable                           Units Coefficient           CI.95    p-va
#>            (Intercept)                                      136.87  [34.96;238.79]   0.008
#>                 gender                          Female         Ref
#>                                                   Male      -13.06  [-16.60;-9.53]    < 1e
#>                    age                                       0.35     [0.24;0.46]    < 1e
#>                   born Born in 50 US states or Washingt         Ref
#>                                                 Others        7.88    [4.06;11.69]    < 1e
#>                   race                           Black         Ref
#>                                                Hispanic       -5.79  [-10.34;-1.24]   0.012
#>                                                   Other       -4.88   [-10.33;0.57]   0.079
#>                                                   White       -0.85    [-5.02;3.33]   0.690
#>              education                         College         Ref
#>                                             High.School        2.85    [-0.32;6.02]   0.078
#>                                                  School       -2.45    [-8.49;3.60]   0.427
```

```
#>                 married                          Married         Ref
#>                                            Never.married       -5.74   [-9.65;-1.83]   0.004
#>                                        Previously.married        0.34    [-3.52;4.20]   0.861
#>                  income                            <25k         Ref
#>                                          Between.25kto54k       -0.87    [-4.77;3.03]   0.663
#>                                          Between.55kto99k        2.46    [-1.79;6.71]   0.256
#>                                                  Over100k        2.63    [-2.07;7.32]   0.272
#>             diastolicBP                                          0.37     [0.25;0.50]    < 1e
#>              systolicBP                                          0.03    [-0.07;0.13]   0.544
#>                     bmi                                         -0.31   [-0.54;-0.08]   0.009
#>           triglycerides                                          0.12     [0.11;0.14]    < 1e
#>               uric.acid                                          1.36     [0.16;2.55]   0.025
#>                 protein                                          4.77   [-0.98;10.51]   0.104
#>               bilirubin                                         -6.06  [-11.14;-0.98]   0.019
#>              phosphorus                                         -0.08    [-2.71;2.55]   0.954
#>                  sodium                                         -1.03   [-1.71;-0.35]   0.003
#>               potassium                                          0.89    [-3.58;5.37]   0.695
#>                globulin                                         -2.20    [-8.15;3.75]   0.469
#>                 calcium                                         12.20    [7.16;17.25]    < 1e
#>           physical.work                              No         Ref
#>                                                     Yes       -0.44    [-3.68;2.80]   0.790
#>  physical.recreational                              No         Ref
#>                                                     Yes        1.24    [-2.03;4.51]   0.457
#>                diabetes                              No         Ref
#>                                                     Yes      -19.03 [-23.26;-14.80]    < 1e

# check if there is still any problematic variable
# with high collinearity problem
collinearity <- ols_vif_tol(fit4)
collinearity[collinearity$VIF>4,]
#> [1] Variables Tolerance VIF
#> <0 rows> (or 0-length row.names)
```

Save data

```
save.image(file = "Data/predictivefactors/cholesterolNHANES15part1.RData")
```

Binary outcome

We focus on statistical analysis and modeling of a binary outcome (cholesterol level) that is categorized as either "healthy" or "unhealthy."

Explore relationships for binary outcome variable

```
#> Loading required package: Hmisc
#> Loading required package: lattice
#> Loading required package: survival
#> Loading required package: Formula
#> Loading required package: ggplot2
#>
#> Attaching package: 'Hmisc'
#> The following objects are masked from 'package:base':
#>
#>     format.pval, units
#> Loading required package: SparseM
#>
#> Attaching package: 'SparseM'
#> The following object is masked from 'package:base':
#>
#>     backsolve
#> Loading required package: prodlim
#> Loading required package: carData
#>
#> Attaching package: 'car'
#> The following objects are masked from 'package:rms':
#>
#>     Predict, vif
#> Type 'citation("pROC")' for a citation.
#>
#> Attaching package: 'pROC'
```

```
#> The following objects are masked from 'package:stats':
#>
#>     cov, smooth, var
```

Load data

```r
load(file = "Data/predictivefactors/cholesterolNHANES15part1.RData")
```

Creating binary variable

Binary categorization: The cholesterol variable is converted into a binary outcome ("healthy" or "unhealthy") using the `ifelse` function based on a threshold value of 200.

Re-leveling: The reference category for the binary variable is changed to "unhealthy."

```r
# Binary variable
analytic3$cholesterol.bin <- ifelse(analytic3$cholesterol < 200, "healthy", "unhealthy")
table(analytic3$cholesterol.bin)
#>
#>   healthy unhealthy
#>      1586      1046

# Changing the reference category
analytic3$cholesterol.bin <- as.factor(analytic3$cholesterol.bin)
analytic3$cholesterol.bin <- relevel(analytic3$cholesterol.bin, ref = "unhealthy")
table(analytic3$cholesterol.bin)
#>
#> unhealthy   healthy
#>      1046      1586
```

Modelling data

A logistic regression is fitted to predict the binary cholesterol outcome from multiple predictor variables.

> **Tip**
>
> We use the `glm` function to run generalized linear models. The default family is gaussian with identity link. Setting binomial family with logit link (logit link is default for binomial family) means fitting logistic regression.

```r
# Regression model
formula5x <- as.formula("cholesterol.bin~gender + age + born +
             race + education + married +
             income + diastolicBP + systolicBP +
             bmi + bodyweight + bodyheight + waist +
             triglycerides + uric.acid +
             protein + bilirubin + phosphorus + sodium + potassium +
             globulin + calcium + physical.work + physical.recreational +
             diabetes")

# Summary
fit5x <- glm(formula5x, family = binomial(), data = analytic3)
publish(fit5x)
#>               Variable                           Units OddsRatio       CI.95    p-value
#>                 gender                          Female       Ref
#>                                                   Male      1.68 [1.27;2.23]   0.000313
#>                    age                                     0.97 [0.97;0.98]    < 1e-04
#>                   born Born in 50 US states or Washingt       Ref
#>                                                 Others      0.69 [0.54;0.88]   0.002636
#>                   race                           Black       Ref
#>                                                Hispanic      1.29 [0.95;1.75]   0.107104
#>                                                   Other      1.24 [0.87;1.79]   0.234062
#>                                                   White      1.10 [0.83;1.44]   0.505147
#>              education                         College       Ref
#>                                             High.School      0.84 [0.68;1.02]   0.082168
#>                                                  School      1.23 [0.84;1.82]   0.292070
#>                married                         Married       Ref
#>                                           Never.married      1.29 [1.00;1.67]   0.052969
#>                                       Previously.married      0.89 [0.70;1.13]   0.345688
#>                 income                            <25k       Ref
#>                                         Between.25kto54k      1.01 [0.78;1.29]   0.957537
#>                                         Between.55kto99k      0.90 [0.69;1.19]   0.462854
#>                                                 Over100k      0.90 [0.66;1.21]   0.472137
```

```
#>            diastolicBP                                    0.98 [0.97;0.98]    < 1e-04
#>             systolicBP                                    1.01 [1.00;1.01]   0.029513
#>                    bmi                                    1.11 [0.99;1.24]   0.065627
#>             bodyweight                                    0.96 [0.92;1.00]   0.045338
#>             bodyheight                                    1.05 [1.00;1.09]   0.030995
#>                  waist                                    1.01 [0.99;1.02]   0.464825
#>          triglycerides                                    0.99 [0.99;0.99]    < 1e-04
#>              uric.acid                                    0.96 [0.89;1.03]   0.273792
#>                protein                                    0.61 [0.42;0.89]   0.009192
#>              bilirubin                                    1.19 [0.86;1.66]   0.292632
#>             phosphorus                                    0.96 [0.81;1.13]   0.610931
#>                 sodium                                    1.06 [1.02;1.11]   0.007980
#>              potassium                                    0.95 [0.71;1.26]   0.729218
#>               globulin                                    1.38 [0.94;2.01]   0.101667
#>                calcium                                    0.64 [0.47;0.89]   0.007026
#>          physical.work                              No     Ref
#>                                                     Yes    0.91 [0.74;1.12]   0.392539
#>  physical.recreational                              No     Ref
#>                                                     Yes    1.05 [0.85;1.29]   0.681388
#>               diabetes                              No     Ref
#>                                                     Yes    2.68 [2.02;3.56]    < 1e-04

# VIF
car::vif(fit5x)
#>                              GVIF Df GVIF^(1/(2*Df))
#> gender                   2.735258  1        1.653862
#> age                      2.121098  1        1.456399
#> born                     1.664094  1        1.289998
#> race                     2.585539  3        1.171544
#> education                1.458430  2        1.098933
#> married                  1.432595  2        1.094034
#> income                   1.426911  3        1.061043
#> diastolicBP              1.297308  1        1.138994
#> systolicBP               1.614374  1        1.270580
#> bmi                     81.928815  1        9.051454
#> bodyweight             103.125772  1       10.155086
#> bodyheight              22.647853  1        4.758976
#> waist                   11.493710  1        3.390237
#> triglycerides            1.258340  1        1.121758
#> uric.acid                1.636512  1        1.279262
```

```
#> protein                 3.684816  1        1.919587
#> bilirubin               1.186181  1        1.089119
#> phosphorus              1.117915  1        1.057315
#> sodium                  1.123193  1        1.059808
#> potassium               1.181358  1        1.086903
#> globulin                3.427401  1        1.851324
#> calcium                 1.543019  1        1.242183
#> physical.work           1.090958  1        1.044490
#> physical.recreational   1.218558  1        1.103883
#> diabetes                1.212365  1        1.101074
```

AUC

The Area Under the Receiver Operating Characteristic (ROC) Curve (AUC) is calculated to assess the model's predictive performance in terms of discrimination. The AUC would tell how much the model is capable of distinguishing between healthy and unhealthy levels.

Let us measure the accuracy for classification models `fit5x`.

> **Tip**
>
> We can use the `roc` function to build a ROC curve and `auc` function to calculate the AUC (are under the ROC curve) value.

```r
require(pROC)
pred.y <- predict(fit5x, type = "response")
rocobj <- roc(analytic3$cholesterol.bin, pred.y)
#> Setting levels: control = unhealthy, case = healthy
#> Setting direction: controls < cases
rocobj
#>
#> Call:
#> roc.default(response = analytic3$cholesterol.bin, predictor = pred.y)
#>
#> Data: pred.y in 1046 controls (analytic3$cholesterol.bin unhealthy) < 1586 cases (analytic3
#> Area under the curve: 0.7411
```

```r

auc(rocobj)
#> Area under the curve: 0.7411
```

Re-modelling

Let us re-fit the model and measure the AUC. VIF is calculated again for this new model.

```r
formula5 <- as.formula("cholesterol.bin~gender + age + born +
             race + education + married +
             income + diastolicBP + systolicBP +
             bmi +
             triglycerides + uric.acid +
             protein + bilirubin + phosphorus + sodium + potassium +
             globulin + calcium + physical.work + physical.recreational +
             diabetes")
fit5 <- glm(formula5, family = binomial(), data = analytic3)
publish(fit5)
#>               Variable                           Units OddsRatio       CI.95    p-value
#>                 gender                          Female       Ref
#>                                                   Male      1.86 [1.48;2.33]    < 1e-04
#>                    age                                    0.97 [0.97;0.98]    < 1e-04
#>                   born Born in 50 US states or Washingt       Ref
#>                                                  Others      0.67 [0.52;0.85]   0.001233
#>                   race                           Black       Ref
#>                                                Hispanic      1.26 [0.94;1.69]   0.128165
#>                                                   Other      1.21 [0.85;1.73]   0.284083
#>                                                   White      1.11 [0.85;1.45]   0.460652
#>              education                         College       Ref
#>                                             High.School      0.84 [0.68;1.02]   0.083806
#>                                                  School      1.22 [0.83;1.80]   0.305514
#>                married                         Married       Ref
#>                                           Never.married      1.29 [1.00;1.68]   0.049983
#>                                       Previously.married      0.89 [0.70;1.13]   0.339401
#>                 income                            <25k       Ref
#>                                         Between.25kto54k      1.00 [0.78;1.28]   0.999445
#>                                         Between.55kto99k      0.90 [0.68;1.17]   0.425427
#>                                                 Over100k      0.89 [0.66;1.20]   0.447012
```

```
#>            diastolicBP                                    0.98 [0.97;0.99]    < 1e-04
#>             systolicBP                                    1.01 [1.00;1.01]   0.042769
#>                    bmi                                    1.01 [0.99;1.02]   0.496430
#>          triglycerides                                    0.99 [0.99;0.99]    < 1e-04
#>              uric.acid                                    0.96 [0.89;1.03]   0.242942
#>                protein                                    0.62 [0.43;0.89]   0.010343
#>              bilirubin                                    1.24 [0.89;1.72]   0.203993
#>             phosphorus                                    0.95 [0.80;1.12]   0.539847
#>                 sodium                                    1.06 [1.02;1.11]   0.006777
#>              potassium                                    0.96 [0.72;1.28]   0.790080
#>               globulin                                    1.37 [0.94;2.00]   0.102430
#>                calcium                                    0.64 [0.46;0.88]   0.005772
#>          physical.work                              No     Ref
#>                                                     Yes    0.91 [0.74;1.12]   0.382281
#>  physical.recreational                              No     Ref
#>                                                     Yes    1.04 [0.85;1.29]   0.682962
#>               diabetes                              No     Ref
#>                                                     Yes    2.69 [2.03;3.57]    < 1e-04

# VIF
car::vif(fit5)
#>                          GVIF Df GVIF^(1/(2*Df))
#> gender                1.749947  1        1.322856
#> age                   1.850160  1        1.360206
#> born                  1.640947  1        1.280994
#> race                  2.345460  3        1.152669
#> education             1.430721  2        1.093676
#> married               1.432015  2        1.093923
#> income                1.409064  3        1.058819
#> diastolicBP           1.289411  1        1.135523
#> systolicBP            1.605248  1        1.266984
#> bmi                   1.477795  1        1.215646
#> triglycerides         1.246395  1        1.116421
#> uric.acid             1.624039  1        1.274378
#> protein               3.648367  1        1.910070
#> bilirubin             1.177643  1        1.085193
#> phosphorus            1.114298  1        1.055603
#> sodium                1.117463  1        1.057101
#> potassium             1.176914  1        1.084857
#> globulin              3.395946  1        1.842809
```

```
#> calcium               1.542486  1        1.241969
#> physical.work         1.089742  1        1.043907
#> physical.recreational 1.197719  1        1.094404
#> diabetes              1.200402  1        1.095629
```

The AUC for this new model is also calculated.

```
#### AUC
pred.y <- predict(fit5, type = "response")
rocobj <- roc(analytic3$cholesterol.bin, pred.y)
#> Setting levels: control = unhealthy, case = healthy
#> Setting direction: controls < cases
rocobj
#>
#> Call:
#> roc.default(response = analytic3$cholesterol.bin, predictor = pred.y)
#>
#> Data: pred.y in 1046 controls (analytic3$cholesterol.bin unhealthy) < 1586 cases (analytic3
#> Area under the curve: 0.7406
auc(rocobj)
#> Area under the curve: 0.7406
```

Save data

```
save.image(file = "Data/predictivefactors/cholesterolNHANES15part2.RData")
```

References

Overfitting and performance

The following tutorial extends the work from the previous lab and focuses on understanding overfitting, evaluating performance, and function writing in the context of linear modeling for a continuous outcome variable, cholesterol levels.

```
# Load required packages
library(caret)
library(knitr)
library(Publish)
library(car)
```

Load data

Load the data saved at the end of previous part of the lab.

```
load(file="Data/predictivefactors/cholesterolNHANES15part2.RData")
```

Now we will fit the final model that we decided at the end of previous part of the lab.

```
formula4 <- as.formula("cholesterol~gender + age + born +
             race + education + married +
             income + diastolicBP + systolicBP +
             bmi + triglycerides + uric.acid +
             protein + bilirubin + phosphorus + sodium + potassium +
             globulin + calcium + physical.work +
             physical.recreational + diabetes")
formula4
#> cholesterol ~ gender + age + born + race + education + married +
#>     income + diastolicBP + systolicBP + bmi + triglycerides +
#>     uric.acid + protein + bilirubin + phosphorus + sodium + potassium +
```

```
#>     globulin + calcium + physical.work + physical.recreational +
#>     diabetes
fit4 <- lm(formula4, data = analytic3)
summary(fit4)
#>
#> Call:
#> lm(formula = formula4, data = analytic3)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -115.465  -23.695   -2.598   20.017  177.264
#>
#> Coefficients:
#>                            Estimate Std. Error t value Pr(>|t|)
#> (Intercept)               136.871606  51.998527   2.632  0.00853 **
#> genderMale                -13.064857   1.802099  -7.250 5.48e-13 ***
#> age                         0.351838   0.056116   6.270 4.22e-10 ***
#> bornOthers                  7.877420   1.947498   4.045 5.39e-05 ***
#> raceHispanic               -5.790547   2.323010  -2.493  0.01274 *
#> raceOther                  -4.879882   2.781673  -1.754  0.07950 .
#> raceWhite                  -0.847635   2.130149  -0.398  0.69072
#> educationHigh.School        2.851633   1.617435   1.763  0.07801 .
#> educationSchool            -2.446765   3.084409  -0.793  0.42769
#> marriedNever.married       -5.739509   1.997152  -2.874  0.00409 **
#> marriedPreviously.married   0.342206   1.968165   0.174  0.86198
#> incomeBetween.25kto54k     -0.867063   1.990253  -0.436  0.66312
#> incomeBetween.55kto99k      2.462130   2.169757   1.135  0.25658
#> incomeOver100k              2.626046   2.394560   1.097  0.27289
#> diastolicBP                 0.374971   0.062238   6.025 1.93e-09 ***
#> systolicBP                  0.029976   0.049515   0.605  0.54497
#> bmi                        -0.309530   0.118927  -2.603  0.00930 **
#> triglycerides               0.124806   0.006427  19.419  < 2e-16 ***
#> uric.acid                   1.357242   0.609012   2.229  0.02593 *
#> protein                     4.767008   2.931636   1.626  0.10406
#> bilirubin                  -6.060791   2.593508  -2.337  0.01952 *
#> phosphorus                 -0.076472   1.341957  -0.057  0.95456
#> sodium                     -1.026686   0.347679  -2.953  0.00318 **
#> potassium                   0.893507   2.283488   0.391  0.69561
#> globulin                   -2.198037   3.036091  -0.724  0.46915
#> calcium                    12.202366   2.574400   4.740 2.25e-06 ***
```

```
#> physical.workYes           -0.439108   1.651078  -0.266  0.79030
#> physical.recreationalYes    1.238756   1.667670   0.743  0.45767
#> diabetesYes               -19.032748   2.158825  -8.816  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 35.22 on 2603 degrees of freedom
#> Multiple R-squared:  0.2415, Adjusted R-squared:  0.2334
#> F-statistic: 29.61 on 28 and 2603 DF,  p-value: < 2.2e-16
```

Design Matrix

Expands factors to a set of dummy variables.

> **Tip**
>
> We can use the `model.matrix` function to construct a design/model matrix, such as expand factor variables to a matrix of dummy variable

The dimensions of the model matrix are obtained, and the total number of model parameters (`p`) is calculated.

```r
head(model.matrix(fit4))
#>    (Intercept) genderMale age bornOthers raceHispanic raceOther raceWhite
#> 1            1          1  62          0            0         0         1
#> 2            1          1  53          1            0         0         1
#> 4            1          0  56          0            0         0         1
#> 5            1          0  42          0            0         0         0
#> 10           1          1  22          0            0         0         0
#> 11           1          0  32          1            1         0         0
#>    educationHigh.School educationSchool marriedNever.married
#> 1                     0               0                    0
#> 2                     1               0                    0
#> 4                     0               0                    0
#> 5                     0               0                    0
#> 10                    0               0                    1
#> 11                    0               0                    0
#>    marriedPreviously.married incomeBetween.25kto54k incomeBetween.55kto99k
```

```
#> 1                          0                      0                      1
#> 2                          1                      0                      0
#> 4                          0                      0                      1
#> 5                          1                      1                      0
#> 10                         0                      1                      0
#> 11                         0                      1                      0
#>    incomeOver100k diastolicBP systolicBP  bmi triglycerides uric.acid protein
#> 1               0          70        128 27.8           158       4.2     7.5
#> 2               0          88        146 30.8           170       7.0     7.4
#> 4               0          72        132 42.4            93       5.4     6.1
#> 5               0          70        100 20.3            52       3.3     7.7
#> 10              0          70        110 28.0            77       6.0     7.4
#> 11              0          70        120 28.2           295       5.2     7.4
#>    bilirubin phosphorus sodium potassium globulin calcium physical.workYes
#> 1        0.5        4.7    136      4.30      2.9     9.8                0
#> 2        0.6        4.4    140      4.55      2.9     9.8                0
#> 4        0.3        3.8    141      4.08      2.3     8.9                0
#> 5        0.3        3.2    136      3.50      3.4     9.3                0
#> 10       0.2        5.3    139      4.16      3.0     9.3                0
#> 11       0.4        3.1    138      4.31      2.9    10.3                0
#>    physical.recreationalYes diabetesYes
#> 1                         0           1
#> 2                         0           0
#> 4                         0           0
#> 5                         0           0
#> 10                        1           0
#> 11                        0           0

# Dimension of the model matrix
dim(model.matrix(fit4))
#> [1] 2632   29

# Number of parameters = intercept + slopes
p <- dim(model.matrix(fit4))[2]
p
#> [1] 29
```

Check prediction

The observed and predicted cholesterol values are summarized.

```
obs.y <- analytic3$cholesterol
summary(obs.y)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    81.0   163.0   189.0   191.5   216.0   362.0

# Predict the above fit on analytic3 data
pred.y <- predict(fit4, analytic3)
summary(pred.y)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   136.3   178.2   189.4   191.5   202.4   337.6
n <- length(pred.y)
n
#> [1] 2632
plot(obs.y,pred.y)
lines(lowess(obs.y,pred.y), col = "red")
```

```

# Prediction on a new data: fictitious.data
str(fictitious.data)
#> 'data.frame':    4121 obs. of  33 variables:
```

```
#>  $ ID                   : num  83732 83733 83734 83735 83736 ...
#>  $ gender               : chr  "Male" "Male" "Male" "Female" ...
#>  $ age                  : num  62 53 78 56 42 72 22 32 56 46 ...
#>  $ born                 : chr  "Born in 50 US states or Washingt" "Others" "Born in 50 US s
#>  $ race                 : chr  "White" "White" "White" "White" ...
#>  $ education            : chr  "College" "High.School" "High.School" "College" ...
#>  $ married              : chr  "Married" "Previously.married" "Married" "Married" ...
#>  $ income               : chr  "Between.55kto99k" "<25k" "<25k" "Between.55kto99k" ...
#>  $ weight               : num  135630 25282 12576 102079 18235 ...
#>  $ psu                  : num  1 1 1 1 2 1 2 1 2 1 ...
#>  $ strata               : num  125 125 131 131 126 128 128 125 126 121 ...
#>  $ diastolicBP          : num  70 88 46 72 70 58 70 70 116 94 ...
#>  $ systolicBP           : num  128 146 138 132 100 116 110 120 178 144 ...
#>  $ bodyweight           : num  94.8 90.4 83.4 109.8 55.2 ...
#>  $ bodyheight           : num  184 171 170 161 165 ...
#>  $ bmi                  : num  27.8 30.8 28.8 42.4 20.3 28.6 28 28.2 33.6 27.6 ...
#>  $ waist                : num  101.1 107.9 116.5 110.1 80.4 ...
#>  $ smoke                : chr  "Not.at.all" "Every.day" "Not.at.all" "Not.at.all" ...
#>  $ alcohol              : num  1 6 0 1 1 0 8 1 0 1 ...
#>  $ cholesterol          : num  173 265 229 174 204 190 164 190 145 242 ...
#>  $ cholesterolM2        : num  4.47 6.85 5.92 4.5 5.28 4.91 4.24 4.91 3.75 6.26 ...
#>  $ triglycerides        : num  158 170 299 93 52 52 77 295 121 497 ...
#>  $ uric.acid            : num  4.2 7 7.3 5.4 3.3 4.9 6 5.2 4.8 6.5 ...
#>  $ protein              : num  7.5 7.4 7.3 6.1 7.7 7.1 7.4 7.4 6.9 6.8 ...
#>  $ bilirubin            : num  0.5 0.6 0.5 0.3 0.3 0.5 0.2 0.4 0.4 0.5 ...
#>  $ phosphorus           : num  4.7 4.4 3.6 3.8 3.2 3.7 5.3 3.1 4.1 3.6 ...
#>  $ sodium               : num  136 140 140 141 136 140 139 138 140 138 ...
#>  $ potassium            : num  4.3 4.55 4.7 4.08 3.5 4.2 4.16 4.31 4.5 4.27 ...
#>  $ globulin             : num  2.9 2.9 2.8 2.3 3.4 3 3 2.9 2.9 2.6 ...
#>  $ calcium              : num  9.8 9.8 9.7 8.9 9.3 9.3 9.3 10.3 9.5 9.3 ...
#>  $ physical.work        : chr  "No" "No" "No" "No" ...
#>  $ physical.recreational: chr  "No" "No" "No" "No" ...
#>  $ diabetes             : chr  "Yes" "No" "Yes" "No" ...
#>  - attr(*, "na.action")= 'omit' Named int [1:885] 16 30 39 48 50 58 61 65 67 68 ...
#>   ..- attr(*, "names")= chr [1:885] "27" "68" "90" "112" ...
pred.y.new1 <- predict(fit4, fictitious.data)
summary(pred.y.new1)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   128.7   178.9   190.6   192.5   203.3   557.4
```

Measuring prediction error

Continuous outcomes

R2

The Sum of Squares of Errors (SSE) and the Total Sum of Squares (SST) are calculated. The proportion of variance explained by the model is then calculated as R2.

See Wikipedia (2023a)

```r
# Find SSE
SSE <- sum( (obs.y - pred.y)^2 )
SSE
#> [1] 3228460

# Find SST
mean.obs.y <- mean(obs.y)
SST <- sum( (obs.y - mean.obs.y)^2 )
SST
#> [1] 4256586

# Find R2
R.2 <- 1- SSE/SST
R.2
#> [1] 0.2415378

require(caret)
R2(pred.y, obs.y)
#> [1] 0.2415378
```

RMSE

The Root Mean Square Error is calculated to measure the average magnitude of the errors between predicted and observed values.

See Wikipedia (2023d)

```r
# Find RMSE
Rmse <- sqrt(SSE/(n-p))
Rmse
#> [1] 35.21767

RMSE(pred.y, obs.y)
#> [1] 35.02311
```

Adj R2

It provides a measure of how well the model generalizes and adjusts R2 based on the number of predictors.

See Wikipedia (2023a)

```r
# Find adj R2
adjR2 <- 1-(1-R.2)*((n-1)/(n-p))
adjR2
#> [1] 0.2333791
```

Writing function

Syntax for Writing Functions

```r
func_name <- function (argument) {
  A statement or multiple lines of statements
  return(output)
}
```

Example of a simple function

```r
f1 <- function(a,b){
  result <- a + b
  return(result)
}
f1(a=1,b=3)
```

```r
#> [1] 4
f1(a=1,b=6)
#> [1] 7
# setting default values
f1 <- function(a=1,b=1){
  result <- a + b
  return(result)
}
f1()
#> [1] 2
f1(b = 10)
#> [1] 11
```

A bit more complicated

```r
# one argument
model.fit <- function(data.for.fitting){
  formulax <- as.formula("cholesterol~gender + age + born")
  fitx <- lm(formulax, data = data.for.fitting)
  result <- coef(fitx)
  return(result)
}
model.fit(data.for.fitting=analytic)
#> (Intercept)  genderMale         age  bornOthers
#> 184.3131838  -7.8095595   0.2225745  11.1557140
model.fit(data.for.fitting=analytic3)
#> (Intercept)  genderMale         age  bornOthers
#> 176.1286576  -4.8256829   0.3375009   7.7186190
```

```r
# adding one more argument: digits
model.fit <- function(data.for.fitting, digits=2){
  formulax <- as.formula("cholesterol~gender + age + born")
  fitx <- lm(formulax, data = data.for.fitting)
  result <- coef(fitx)
  result <- round(result,digits)
  return(result)
}
model.fit(data.for.fitting=analytic)
```

```
#> (Intercept)  genderMale         age  bornOthers
#>      184.31       -7.81        0.22       11.16
model.fit(data.for.fitting=analytic3)
#> (Intercept)  genderMale         age  bornOthers
#>      176.13       -4.83        0.34        7.72
```

Function that gives performance measures

let us create a function that will give us the performance measures:

```
perform <- function(new.data,
                    model.fit,model.formula=NULL,
                    y.name = "Y",
                    digits=3){
  # data dimension
  p <- dim(model.matrix(model.fit))[2]

  # predicted value
  pred.y <- predict(model.fit, new.data)

  # sample size
  n <- length(pred.y)

  # outcome
  new.data.y <- as.numeric(new.data[,y.name])

  # R2
  R2 <- caret:::R2(pred.y, new.data.y)

  # adj R2 using alternate formula
  df.residual <- n-p
  adjR2 <- 1-(1-R2)*((n-1)/df.residual)

  # RMSE
  RMSE <-  caret:::RMSE(pred.y, new.data.y)

  # combine all of the results
  res <- round(cbind(n,p,R2,adjR2,RMSE),digits)
```

```r

  # returning object
  return(res)
}
perform(new.data = analytic3, y.name = "cholesterol", model.fit = fit4)
#>         n  p    R2 adjR2   RMSE
#> [1,] 2632 29 0.242 0.233 35.023
```

References

Data Splitting

This tutorial is focused on a crucial aspect of model building: splitting your data into training and test sets to avoid overfitting. Overfitting occurs when your model learns the noise in the data rather than the underlying trend. As a result, the model performs well on the training data but poorly on new, unseen data. To mitigate this, you often split your data.

Load data anf files

Initially, several libraries are loaded to facilitate data manipulation and analysis.

```
# Load required packages
library(caret)
library(knitr)
library(Publish)
library(car)
library(DescTools)
```

Then, previously saved dataset related to cholesterol and other factors is loaded for further use.

```
load(file="Data/predictivefactors/cholesterolNHANES15part2.RData")
```

Data splitting to avoid model overfitting

You start by setting a random seed to ensure that the random splitting of data is reproducible. A specified function is then used to partition the data, taking as arguments the outcome variable (cholesterol level in this case) and the percentage of

data that you want to allocate to the training set (70% in this example).

> **Tip**
>
> We can use the `createDataPartition` function to split a dataset into training and testing datasets. The function will return the row indices that should go into the training set. These indices are stored in a variable, and its dimensions are displayed to provide an understanding of the size of the training set that will be created. Additionally, you can calculate what 70% of your entire dataset would look like to verify the approximation of the training data size, as well as what the remaining 30% (for the test set) would look like.

```r
# Using a seed to randomize in a reproducible way
set.seed(123)
split <- createDataPartition(y = analytic3$cholesterol, p = 0.7, list = FALSE)
str(split)
#>  int [1:1844, 1] 3 4 5 8 9 13 14 16 20 21 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : NULL
#>   ..$ : chr "Resample1"
dim(split)
#> [1] 1844    1

# Approximate train data
dim(analytic3)*.7
#> [1] 1842.4   24.5

# Approximate test data
dim(analytic3)*(1-.7)
#> [1] 789.6  10.5
```

Split the data

After determining how to partition the data, the next step is actually creating the training and test datasets. The indices are

used to subset the original dataset into these two new datasets. The dimensions of each dataset are displayed to confirm their sizes.

```
# Create train data
train.data <- analytic3[split,]
dim(train.data)
#> [1] 1844   35

# Create test data
test.data <- analytic3[-split,]
dim(test.data)
#> [1] 788  35
```

Our next task is to fit the model (e.g., linear regression) on the training set and evaluate the performance on the test set.

Train the model

Once the training dataset is created, you can proceed to train the model using the training data. A previously defined formula containing the predictor variables is used in a linear regression model. After fitting the model, a summary is generated to display key statistics that help in evaluating the model's performance.

```
formula4
#> cholesterol ~ gender + age + born + race + education + married +
#>     income + diastolicBP + systolicBP + bmi + triglycerides +
#>     uric.acid + protein + bilirubin + phosphorus + sodium + potassium +
#>     globulin + calcium + physical.work + physical.recreational +
#>     diabetes
fit4.train1 <- lm(formula4, data = train.data)
summary(fit4.train1)
#>
#> Call:
#> lm(formula = formula4, data = train.data)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
```

```
#> -91.973 -23.719  -1.563  20.586 178.542
#>
#> Coefficients:
#>                            Estimate Std. Error t value Pr(>|t|)
#> (Intercept)               72.716792  59.916086   1.214  0.22504
#> genderMale               -11.293629   2.136545  -5.286 1.40e-07 ***
#> age                        0.306235   0.066376   4.614 4.23e-06 ***
#> bornOthers                 7.220858   2.300658   3.139  0.00172 **
#> raceHispanic              -6.727473   2.709718  -2.483  0.01313 *
#> raceOther                 -4.865771   3.237066  -1.503  0.13298
#> raceWhite                 -1.468522   2.494981  -0.589  0.55621
#> educationHigh.School       1.626097   1.920289   0.847  0.39722
#> educationSchool           -4.853095   3.585185  -1.354  0.17602
#> marriedNever.married      -5.298265   2.332033  -2.272  0.02321 *
#> marriedPreviously.married  1.202448   2.305191   0.522  0.60199
#> incomeBetween.25kto54k    -1.736495   2.360385  -0.736  0.46202
#> incomeBetween.55kto99k     0.170505   2.565896   0.066  0.94703
#> incomeOver100k             1.712359   2.860226   0.599  0.54946
#> diastolicBP                0.355813   0.074380   4.784 1.86e-06 ***
#> systolicBP                 0.037464   0.059848   0.626  0.53140
#> bmi                       -0.282881   0.139160  -2.033  0.04222 *
#> triglycerides              0.123797   0.007613  16.261  < 2e-16 ***
#> uric.acid                  1.006499   0.712871   1.412  0.15815
#> protein                    1.721623   3.468969   0.496  0.61975
#> bilirubin                 -6.143411   3.006858  -2.043  0.04118 *
#> phosphorus                 0.093824   1.575489   0.060  0.95252
#> sodium                    -0.604286   0.400694  -1.508  0.13170
#> potassium                 -0.583525   2.715189  -0.215  0.82986
#> globulin                  -0.278970   3.614404  -0.077  0.93849
#> calcium                   15.679677   3.054968   5.133 3.17e-07 ***
#> physical.workYes          -1.099540   1.960321  -0.561  0.57494
#> physical.recreationalYes   0.834737   1.953960   0.427  0.66928
#> diabetesYes              -19.932101   2.580138  -7.725 1.83e-14 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 34.68 on 1815 degrees of freedom
#> Multiple R-squared:  0.2433, Adjusted R-squared:  0.2316
#> F-statistic: 20.84 on 28 and 1815 DF,  p-value: < 2.2e-16
```

Extract performance measures

You can use a saved function to measure the performance of the trained model. The function will return performance metrics like R-squared, RMSE, etc. This function is applied not just to the training data but also to the test data, the full dataset, and a separate, fictitious dataset.

> **Tip**
>
> Below we use the `perform` function that we saved to evaluate the model performances

```
perform(new.data = train.data,y.name = "cholesterol", model.fit = fit4.train1)
#>         n  p df.residual     SSE     SST    R2 adjR2  sigma   logLik      AIC
#> [1,] 1844 29        1815 2182509 2884109 0.243 0.232 34.677 -9140.98 18341.96
#>           BIC
#> [1,] 18507.55
perform(new.data = test.data,y.name = "cholesterol", model.fit = fit4.train1)
#>        n  p df.residual     SSE     SST    R2 adjR2  sigma    logLik      AIC
#> [1,] 788 29         759 1057454 1372214 0.229 0.201 37.326 -3955.936 7971.873
#>           BIC
#> [1,] 8111.958
perform(new.data = analytic3,y.name = "cholesterol", model.fit = fit4.train1)
#>         n  p df.residual     SSE     SST    R2 adjR2 sigma    logLik      AIC
#> [1,] 2632 29        2603 3239962 4256586 0.239 0.231 35.28 -13098.82 26257.64
#>           BIC
#> [1,] 26433.91
perform(new.data = fictitious.data,y.name = "cholesterol", model.fit = fit4.train1)
#>         n  p df.residual     SSE     SST    R2 adjR2  sigma    logLik      AIC
#> [1,] 4121 29        4092 5306559 6912485 0.232 0.227 36.011 -20601.92 41263.84
#>           BIC
#> [1,] 41453.55
```

Evaluating the model's performance on the test data provides insights into how well the model will generalize to new, unseen data. Comparing the performance metrics across different datasets can give you a robust view of your model's predictive power and reliability.

For more on model training and tuning, see Kuhn (2023c)

References

Cross-validation

Cross-validation is another important technique used to assess the performance of machine learning models and mitigate the risk of overfitting. This tutorial focuses on k-fold cross-validation as a strategy to obtain a more generalized and robust assessment of the model's performance. It shows both manual calculations for individual folds and an automated approach using the caret package. This ensures that you aren't simply fitting your model well to a specific subset of your data but are achieving good performance in a general sense.

```r
# Load required packages
library(caret)
library(knitr)
library(Publish)
library(car)
library(DescTools)
```

Load data

Load the data saved at the end of previous part of the lab.

```r
load(file="Data/predictivefactors/cholesterolNHANES15part2.RData")
```

k-fold cross-vaildation

We can set the number of folds to 5 (k = 5). A random seed is used for reproducibility. We use the function `createFolds` to create the folds. The data is divided based on the cholesterol levels, with each fold having approximately equal numbers of data points. The resulting structure contains training indices for each fold.

See Wikipedia (2023b)

We can also examine the approximate size of training and test sets for each fold. The dimensions are displayed to understand the partitioning, and you can examine the length of indices in each fold to confirm the size of the training sets.

```r
k = 5
dim(analytic3)
#> [1] 2632   35
set.seed(567)

# Create folds (based on the outcome)
folds <- createFolds(analytic3$cholesterol, k = k, list = TRUE,
                     returnTrain = TRUE)
mode(folds)
#> [1] "list"

# Approximate training data size
dim(analytic3)*4/5
#> [1] 2105.6   28.0

# Approximate test data size
dim(analytic3)/5
#> [1] 526.4   7.0

length(folds[[1]])
#> [1] 2105
length(folds[[2]])
#> [1] 2107
length(folds[[3]])
#> [1] 2106
length(folds[[4]])
#> [1] 2105
length(folds[[5]])
#> [1] 2105

str(folds[[1]])
#>  int [1:2105] 1 3 5 6 8 10 11 12 13 14 ...
str(folds[[2]])
#>  int [1:2107] 1 2 3 4 5 6 7 8 9 12 ...
str(folds[[3]])
#>  int [1:2106] 2 4 5 7 8 9 10 11 12 14 ...
```

```r
str(folds[[4]])
#>  int [1:2105] 1 2 3 4 6 7 8 9 10 11 ...
str(folds[[5]])
#>  int [1:2105] 1 2 3 4 5 6 7 9 10 11 ...
```

Calculation for Fold 1

The first fold is used as an example. The indices for the training data in the first fold are extracted and used to subset the main data set into training and test sets for that fold. Then a linear regression model is fitted using the training data, and predictions are made on the test set. The model's performance is evaluated using the same performance function as before.

```r
fold.index <- 1
fold1.train.ids <- folds[[fold.index]]
head(fold1.train.ids)
#> [1]  1  3  5  6  8 10

fold1.train <- analytic3[fold1.train.ids,]
fold1.test <- analytic3[-fold1.train.ids,]
formula4
#> cholesterol ~ gender + age + born + race + education + married +
#>     income + diastolicBP + systolicBP + bmi + triglycerides +
#>     uric.acid + protein + bilirubin + phosphorus + sodium + potassium +
#>     globulin + calcium + physical.work + physical.recreational +
#>     diabetes

model.fit <- lm(formula4, data = fold1.train)
predictions <- predict(model.fit, newdata = fold1.test)

perform(new.data=fold1.test, y.name = "cholesterol",
        model.fit = model.fit)
#>        n  p df.residual      SSE      SST    R2 adjR2  sigma    logLik      AIC
#> [1,] 527 29         498 637317.5 830983.2 0.233  0.19 35.774 -2618.471 5296.942
#>           BIC
#> [1,] 5424.958
```

Calculation for Fold 2

The same process is repeated for the second fold. This way, you can manually evaluate how the model performs on different subsets of the data, making the performance assessment more robust.

```
fold.index <- 2
fold1.train.ids <- folds[[fold.index]]
head(fold1.train.ids)
#> [1] 1 2 3 4 5 6

fold1.train <- analytic3[fold1.train.ids,]
fold1.test <- analytic3[-fold1.train.ids,]

model.fit <- lm(formula4, data = fold1.train)

predictions <- predict(model.fit, newdata = fold1.test)
perform(new.data=fold1.test, y.name = "cholesterol",
        model.fit = model.fit)
#>        n  p df.residual    SSE      SST    R2 adjR2  sigma    logLik      AIC
#> [1,] 525 29         496 615243 785326.6 0.217 0.172 35.219 -2600.282 5260.564
#>           BIC
#> [1,] 5388.466
```

Using caret package to automate

Instead of manually running the process for each fold, the caret package can be used to automate k-fold cross-validation. A control object is set up specifying that 5-fold cross-validation should be used. Then, the train function from the caret package can be used to fit the linear regression model on each fold.

See Kuhn (2023c)

After fitting, you can access summary results for each fold in the resampling results. This summary provides performance metrics such as R-squared for each fold. You can calculate the mean and standard deviation of these metrics to get an overall sense of the model's performance.

Additionally, an adjusted R-squared can be calculated to consider the number of predictors in the model, giving a more accurate sense of the model's explanatory power when you have multiple predictors.

```r
# Using Caret package
set.seed(567)

# make a 5-fold CV
ctrl<-trainControl(method = "cv",number = 5)

# fit the model with formula = formula4
# use training method lm
fit4.cv<-train(formula4, trControl = ctrl,
               data = analytic3, method = "lm")
fit4.cv
#> Linear Regression
#>
#> 2632 samples
#>   22 predictor
#>
#> No pre-processing
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 2106, 2105, 2106, 2105, 2106
#> Resampling results:
#>
#>   RMSE      Rsquared   MAE
#>   35.62758  0.2194187  27.85731
#>
#> Tuning parameter 'intercept' was held constant at a value of TRUE

# extract results from each test data
summary.res <- fit4.cv$resample
summary.res
#>       RMSE  Rsquared      MAE Resample
#> 1 36.85337 0.2050676 28.48951    Fold1
#> 2 34.99024 0.2660231 27.57746    Fold2
#> 3 35.25756 0.2056371 26.85106    Fold3
#> 4 35.48300 0.1979601 28.44168    Fold4
#> 5 35.55372 0.2224055 27.92683    Fold5
mean(fit4.cv$resample$Rsquared)
```

```r
#> [1] 0.2194187
sd(fit4.cv$resample$Rsquared)
#> [1] 0.02755561

# # extract adj R2
# k <- 5
# p <- 2
# n <- round(nrow(analytic3)/k)
# summary.res$adjR2 <- 1-(1-fit4.cv$resample$Rsquared)*
#  ((n-1)/(n-p))
# summary.res
```

References

Bootstrap

The tutorial is on bootstrapping methods, mainly using R. Bootstrapping is a resampling technique used to estimate the sampling distribution of a statistic by repeatedly sampling, with replacement (after a data point is chosen randomly from the original dataset and included in the sample, it is "replaced" back into the original dataset, making it possible for that same data point to be picked again in the same sampling process), from the observed data points. It is a way to quantify the uncertainty associated with a given estimator or statistical measure, such as the mean, median, variance, or correlation coefficient, among others. Bootstrapping is widely applicable and very straightforward to implement, which has made it a popular choice for statistical inference when analytical solutions are not available or are difficult to derive.

> **!** Important
>
> Bootstrapping is a powerful statistical tool for making inferences by empirically estimating the sampling distribution of a statistic. It is especially useful when the underlying distribution is unknown or when an analytical solution is difficult to obtain.

```r
# Load required packages
library(caret)
library(knitr)
library(Publish)
library(car)
library(DescTools)
```

Load data

Load the data saved at the end of previous part of the lab.

```
load(file="Data/predictivefactors/cholesterolNHANES15part2.RData")
```

Resampling a vector

Here, the document introduces basic resampling of a simple vector. The code creates a new sample using the `sample` function with replacement. It also discusses "out-of-bag" samples which are the samples not chosen during the resampling.

```
fake.data <- 1:5
fake.data
#> [1] 1 2 3 4 5
```

```
resampled.fake.data <- sample(fake.data, size = length(fake.data),
                              replace = TRUE)
resampled.fake.data
#> [1] 4 3 2 5 3

selected.fake.data <- unique(resampled.fake.data)
selected.fake.data
#> [1] 4 3 2 5

fake.data[!(fake.data %in% selected.fake.data)]
#> [1] 1
```

The samples not selected are known as the out-of-bag samples

```
B <- 10
for (i in 1:B){
  new.boot.sample <- sample(fake.data, size = length(fake.data),
                            replace = TRUE)
  print(new.boot.sample)
}
#> [1] 1 2 4 2 1
#> [1] 4 4 2 2 1
```

```
#> [1] 1 1 4 3 2
#> [1] 4 5 3 5 3
#> [1] 2 2 1 3 4
#> [1] 1 1 4 2 4
#> [1] 2 3 4 2 1
#> [1] 4 2 1 3 2
#> [1] 1 4 3 1 4
#> [1] 1 5 5 3 4
```

Calculating SD of a statistics

We introduce the concept of calculating confidence intervals (CIs) using bootstrapping when the distribution of data is not known. It uses resampling to create multiple bootstrap samples, then calculates means and standard deviations (SD) for those samples.

Idea:

- Not sure about what distribution is appropriate to make inference?
- If that is the case, calculating CI is hard.
- resample and get a new bootstrap sample
- calculate a statistic (say, mean) from that sample
- find SD of those statistic (say, means)
- Use those SD to calculate CI

```r
mean(fake.data)
#> [1] 3
B <- 5
resamples <- lapply(1:B, function(i) sample(fake.data,
                                            replace = TRUE))
str(resamples)
#> List of 5
#>  $ : int [1:5] 1 2 4 3 1
#>  $ : int [1:5] 4 1 5 2 2
#>  $ : int [1:5] 2 4 3 2 2
#>  $ : int [1:5] 4 4 4 4 2
#>  $ : int [1:5] 3 2 5 1 4

```

```r
B.means <- sapply(resamples, mean)
B.means
#> [1] 2.2 2.8 2.6 3.6 3.0
mean(B.means)
#> [1] 2.84

# SD of the distribution of means
sd(B.means)
#> [1] 0.5176872
```

```r
mean(fake.data)
#> [1] 3
B <- 200
resamples <- lapply(1:B, function(i) sample(fake.data,
                                            replace = TRUE))
# str(resamples)

B.means <- sapply(resamples, mean)
B.medians <- sapply(resamples, median)
mean(B.means)
#> [1] 3.034

# SD of the distribution of means
sd(B.means)
#> [1] 0.6202966
mean(B.medians)
#> [1] 3.115
hist(B.means)
```

```
# SD of the distribution of medians
sd(B.medians)
#> [1] 0.9780377
hist(B.medians)
```

Resampling a data or matrix

We show how to resample a data frame or a matrix, and how to identify which rows have been selected and which haven't,

introducing the concept of "out-of-bag samples" for matrices.

```r
analytic.mini <- head(analytic)
kable(analytic.mini[,1:3])
```

	ID	gender	age
1	83732	Male	62
2	83733	Male	53
10	83741	Male	22
16	83747	Male	46
19	83750	Male	45
21	83752	Female	30

```r
analytic.boot <- analytic.mini[sample(x = 1:nrow(analytic.mini),
                                      size = nrow(analytic.mini),
                                      replace = TRUE), ]
kable(analytic.boot[,1:3])
```

	ID	gender	age
2	83733	Male	53
19	83750	Male	45
19.1	83750	Male	45
19.2	83750	Male	45
10	83741	Male	22
10.1	83741	Male	22

```r
selected.subjects <- unique(analytic.boot$ID)
selected.subjects
#> [1] 83733 83750 83741

# out-of-bag samples
analytic.mini$ID[!(analytic.mini$ID %in% selected.subjects)]
#> [1] 83732 83747 83752
```

```r
analytic.boot <- analytic.mini[sample(x = 1:nrow(analytic.mini),
                                      size = nrow(analytic.mini),
                                      replace = TRUE), ]
kable(analytic.boot[,1:3])
```

	ID	gender	age
21	83752	Female	30
10	83741	Male	22
19	83750	Male	45
16	83747	Male	46
2	83733	Male	53
21.1	83752	Female	30

```r
selected.subjects <- unique(analytic.boot$ID)
selected.subjects
#> [1] 83752 83741 83750 83747 83733

# out-of-bag samples
analytic.mini$ID[!(analytic.mini$ID %in% selected.subjects)]
#> [1] 83732
```

The caret package / boot

Usually B = 200 or 500 is recommended, but we will do 50 for the lab (to save time). We introduce the `trainControl` and `train` functions from the caret package. It sets up a linear model and demonstrates how bootstrapping can be done to estimate the variability in R-squared, a measure of goodness-of-fit for the model.

```r
set.seed(234)
ctrl<-trainControl(method = "boot", number = 50)
fit4.boot2<-train(formula4, trControl = ctrl,
                  data = analytic3, method = "lm")
fit4.boot2
#> Linear Regression
#>
#> 2632 samples
#>   22 predictor
#>
#> No pre-processing
#> Resampling: Bootstrapped (50 reps)
#> Summary of sample sizes: 2632, 2632, 2632, 2632, 2632, 2632, ...
```

```
#> Resampling results:
#>
#>   RMSE      Rsquared  MAE
#>   35.58231  0.22375   27.77634
#>
#> Tuning parameter 'intercept' was held constant at a value of TRUE

head(fit4.boot2$resample)
#>       RMSE  Rsquared      MAE   Resample
#> 1 35.94822 0.2065802 27.99081 Resample01
#> 2 34.63989 0.2176692 26.94006 Resample02
#> 3 36.09819 0.2067883 27.92833 Resample03
#> 4 35.50718 0.2260809 27.05782 Resample04
#> 5 35.07730 0.2161031 27.21376 Resample05
#> 6 36.87788 0.2504643 28.70333 Resample06
mean(fit4.boot2$resample$Rsquared)
#> [1] 0.22375
sd(fit4.boot2$resample$Rsquared)
#> [1] 0.01693917
```

Method boot632

A specific bootstrapping method called "boot632", which aims to reduce bias but can provide unstable results if the sample size is small. Compared to the original bootstrap method, boot632 addresses the bias that is due to this the sampling with replacement.

See Raschka (2023)

```
ctrl<-trainControl(method = "boot632", number = 50)
fit4.boot2b<-train(formula4, trControl = ctrl,
                   data = analytic3, method = "lm")
fit4.boot2b
#> Linear Regression
#>
#> 2632 samples
#>   22 predictor
#>
#> No pre-processing
```

```
#> Resampling: Bootstrapped (50 reps)
#> Summary of sample sizes: 2632, 2632, 2632, 2632, 2632, 2632, ...
#> Resampling results:
#>
#>   RMSE      Rsquared   MAE
#>   35.33279  0.2277843  27.58945
#>
#> Tuning parameter 'intercept' was held constant at a value of TRUE

head(fit4.boot2b$resample)
#>       RMSE  Rsquared      MAE   Resample
#> 1 35.71916 0.2091238 27.79221 Resample01
#> 2 36.03436 0.2088865 27.87216 Resample02
#> 3 35.54895 0.2196406 27.77477 Resample03
#> 4 34.96061 0.2374635 27.27394 Resample04
#> 5 35.90109 0.2154780 28.11286 Resample05
#> 6 35.84707 0.2395014 27.66055 Resample06
mean(fit4.boot2b$resample$Rsquared)
#> [1] 0.2197801
sd(fit4.boot2b$resample$Rsquared)
#> [1] 0.02259778
```

Method boot632 for stepwise

We discuss the use of stepwise regression models in conjunction with the "boot632" method. It highlights the trade-offs and explains that models could be unstable depending on the data.

A stable model

Bias is reduced with 632 bootstrap, but may provide unstable results with a small samples size.

See Kuhn (2023a)

```
ctrl <- trainControl(method = "boot632", number = 50)
fit4.boot2b<-train(formula4, trControl = ctrl,
                   data = analytic3, method = "lmStepAIC",
                   trace = 0)
fit4.boot2b
```

```
#> Linear Regression with Stepwise Selection
#>
#> 2632 samples
#>   22 predictor
#>
#> No pre-processing
#> Resampling: Bootstrapped (50 reps)
#> Summary of sample sizes: 2632, 2632, 2632, 2632, 2632, 2632, ...
#> Resampling results:
#>
#>   RMSE      Rsquared   MAE
#>   35.34494  0.2293058  27.65063

head(fit4.boot2b$resample)
#>       RMSE  Rsquared      MAE   Resample
#> 1 35.42297 0.2571716 27.94171 Resample01
#> 2 35.84409 0.2347763 28.07948 Resample02
#> 3 35.90088 0.2271463 27.84382 Resample03
#> 4 34.83679 0.2264647 27.53455 Resample04
#> 5 35.17698 0.2093156 27.45790 Resample05
#> 6 35.08153 0.1811553 27.56418 Resample06
mean(fit4.boot2b$resample$Rsquared)
#> [1] 0.2226174
sd(fit4.boot2b$resample$Rsquared)
#> [1] 0.01922833
```

An unstable model

```
ctrl<-trainControl(method = "boot632", number = 50)

# formula3 includes collinear variables
fit4.boot2b<-train(formula3, trControl = ctrl,
                   data = analytic3, method = "lmStepAIC",
                   trace = 0)
fit4.boot2b
#> Linear Regression with Stepwise Selection
#>
#> 2632 samples
```

```
#>   25 predictor
#>
#> No pre-processing
#> Resampling: Bootstrapped (50 reps)
#> Summary of sample sizes: 2632, 2632, 2632, 2632, 2632, 2632, ...
#> Resampling results:
#>
#>   RMSE      Rsquared   MAE
#>   35.39802  0.2287758  27.6471

head(fit4.boot2b$resample)
#>       RMSE  Rsquared      MAE   Resample
#> 1 36.11835 0.2015042 27.87232 Resample01
#> 2 36.05382 0.2255470 28.11558 Resample02
#> 3 36.28309 0.2064684 28.44588 Resample03
#> 4 35.64308 0.2076057 27.79977 Resample04
#> 5 37.05636 0.2151877 28.77376 Resample05
#> 6 34.96517 0.2257338 27.32477 Resample06
mean(fit4.boot2b$resample$Rsquared)
#> [1] 0.2205909
sd(fit4.boot2b$resample$Rsquared)
#> [1] 0.0176326
```

Note that SD should be higher for larger B.

Optimism corrected bootstrap

We discuss a specific type of bootstrap called the "Optimism corrected bootstrap". It's a way to adjust performance metrics for the optimism that is often present when a model is tested on the data used to create it.

Steps:

See Bondarenko and Consulting (2023)

- Fit a model M to entire data D and estimate predictive ability R2.
- Iterate from b=1 to B:
  - Take a resample from the original data, and name it D.star

  - Fit the bootstrap model M.stat to D.star and get predictive ability, R2.boot
  - Use the bootstrap model M.star to get predictive ability on D, R2.fullData

- Optimism Opt is calculated as mean(R2.boot - R2.fullData)
- Calculate optimism corrected performance as R2-Opt.

```r
R2.opt <- function(data, fit, B, y.name = "cholesterol"){
  D <- data
  y.index <- which(names(D)==y.name)

  # M is the model fit to entire data D
  M <- fit
  pred.y <- predict(M, D)
  n <- length(pred.y)
  y <- as.numeric(D[,y.index])

  # estimate predictive ability R2.
  R2.app <- caret:::R2(pred.y, y)

  # create blank vectors to save results
  R2.boot <- vector (mode = "numeric", length = B)
  R2.fullData <- vector (mode = "numeric", length = B)
  opt <- vector (mode = "numeric", length = B)

  # Iterate from b=1 to B
  for(i in 1:B){
    # Take a resample from the original data, and name it D.star
    boot.index <- sample(x=rownames(D), size=nrow(D), replace=TRUE)
    D.star <- D[boot.index,]
    M.star <- lm(formula(M), data = D.star)

    # Fit the bootstrap model M.stat to D.star and get predictive ability, R2.boot
    D.star$pred.y <- predict(M.star, new.data = D.star)
    y.index <- which(names(D.star)==y.name)
    D.star$y <- as.numeric(D.star[,y.index])
    R2.boot[i] <- caret:::R2(D.star$pred.y, D.star$y)

    # Use the bootstrap model M.star to get predictive ability on D, R2_fullData
```

```r
    D$pred.y <- predict(M.star, newdata=D)
    R2.fullData[i] <- caret:::R2(D$pred.y, y)

    # Optimism Opt is calculated as R2.boot - R2.fullData
    opt[i] <- R2.boot[i] - R2.fullData[i]
  }
  boot.res <- round(cbind(R2.boot, R2.fullData,opt),2)
  # Calculate optimism corrected performance as R2- mean(Opt).
  R2.oc <- R2.app - (sum(opt)/B)
  return(list(R2.oc=R2.oc,R2.app=R2.app, boot.res = boot.res))
}

R2x <- R2.opt(data = analytic3, fit4, B=50)
R2x
#> $R2.oc
#> [1] 0.2238703
#>
#> $R2.app
#> [1] 0.2415378
#>
#> $boot.res
#>       R2.boot R2.fullData   opt
#>  [1,]    0.23        0.24 -0.01
#>  [2,]    0.24        0.23  0.01
#>  [3,]    0.26        0.24  0.03
#>  [4,]    0.25        0.23  0.02
#>  [5,]    0.26        0.24  0.02
#>  [6,]    0.26        0.23  0.03
#>  [7,]    0.21        0.24 -0.03
#>  [8,]    0.25        0.23  0.02
#>  [9,]    0.24        0.23  0.01
#> [10,]    0.27        0.23  0.03
#> [11,]    0.25        0.23  0.01
#> [12,]    0.24        0.23  0.01
#> [13,]    0.26        0.23  0.03
#> [14,]    0.25        0.24  0.02
#> [15,]    0.25        0.23  0.02
#> [16,]    0.24        0.23  0.00
#> [17,]    0.25        0.23  0.02
#> [18,]    0.26        0.24  0.03
```

```
#> [19,]    0.24         0.24  0.01
#> [20,]    0.27         0.24  0.03
#> [21,]    0.27         0.24  0.04
#> [22,]    0.26         0.23  0.02
#> [23,]    0.23         0.23  0.00
#> [24,]    0.23         0.23  0.00
#> [25,]    0.26         0.23  0.03
#> [26,]    0.26         0.23  0.03
#> [27,]    0.27         0.23  0.04
#> [28,]    0.27         0.24  0.03
#> [29,]    0.27         0.23  0.04
#> [30,]    0.24         0.23  0.00
#> [31,]    0.25         0.23  0.02
#> [32,]    0.25         0.24  0.02
#> [33,]    0.26         0.24  0.02
#> [34,]    0.23         0.24  0.00
#> [35,]    0.25         0.23  0.02
#> [36,]    0.26         0.23  0.03
#> [37,]    0.26         0.23  0.03
#> [38,]    0.23         0.24  0.00
#> [39,]    0.26         0.23  0.03
#> [40,]    0.27         0.23  0.03
#> [41,]    0.24         0.23  0.01
#> [42,]    0.24         0.24  0.00
#> [43,]    0.28         0.23  0.04
#> [44,]    0.25         0.24  0.02
#> [45,]    0.25         0.23  0.02
#> [46,]    0.26         0.24  0.02
#> [47,]    0.25         0.23  0.02
#> [48,]    0.25         0.23  0.02
#> [49,]    0.25         0.24  0.02
#> [50,]    0.23         0.23 -0.01
```

Binary outcome

Here, bootstrapping and cross-validation are used for a logistic regression model. It calculates the Area Under the Receiver Operating Characteristic Curve (AUC-ROC), a measure for the performance of classification models.

- AUC from Receiver Operating Characteristic (ROC) = Measure of accuracy for classification models.
- AUC = 1 (perfect classification)
- AUC = 0.5 (random classification such as a coin toss)

```r
set.seed(234)
formula5
#> cholesterol.bin ~ gender + age + born + race + education + married +
#>     income + diastolicBP + systolicBP + bmi + triglycerides +
#>     uric.acid + protein + bilirubin + phosphorus + sodium + potassium +
#>     globulin + calcium + physical.work + physical.recreational +
#>     diabetes

# Bootstrap
ctrl<-trainControl(method = "boot",
                   number = 50,
                   classProbs=TRUE,
                   summaryFunction = twoClassSummary)

fit5.boot<-caret::train(formula5,
                        trControl = ctrl,
                        data = analytic3,
                        method = "glm",
                        family="binomial",
                        metric="ROC")
fit5.boot
#> Generalized Linear Model
#>
#> 2632 samples
#>   22 predictor
#>    2 classes: 'unhealthy', 'healthy'
#>
#> No pre-processing
#> Resampling: Bootstrapped (50 reps)
#> Summary of sample sizes: 2632, 2632, 2632, 2632, 2632, 2632, ...
#> Resampling results:
#>
#>   ROC        Sens       Spec
#>   0.7238856  0.4563417  0.8201976
mean(fit5.boot$resample$ROC)
```

```r
#> [1] 0.7238856
sd(fit5.boot$resample$ROC)
#> [1] 0.01166374

# CV
ctrl <- trainControl(method = "cv",
                     number = 5,
                     classProbs = TRUE,
                     summaryFunction = twoClassSummary)

fit5.cv <- train(formula5,
                 trControl = ctrl,
                 data = analytic3,
                 method = "glm",
                 family="binomial",
                 metric="ROC")
fit5.cv
#> Generalized Linear Model
#>
#> 2632 samples
#>   22 predictor
#>    2 classes: 'unhealthy', 'healthy'
#>
#> No pre-processing
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 2106, 2106, 2105, 2105, 2106
#> Resampling results:
#>
#>   ROC        Sens       Spec
#>   0.7291594  0.4512144  0.8253358
fit5.cv$resample
#>         ROC      Sens      Spec Resample
#> 1 0.6956364 0.4593301 0.7823344    Fold1
#> 2 0.7217183 0.4114833 0.8422713    Fold2
#> 3 0.7214511 0.4809524 0.8044164    Fold3
#> 4 0.7383768 0.4354067 0.8427673    Fold4
#> 5 0.7686143 0.4688995 0.8548896    Fold5
mean(fit5.cv$resample$ROC)
#> [1] 0.7291594
sd(fit5.cv$resample$ROC)
```

```
#> [1] 0.02683386
```

Brier Score is another metric for evaluating the performance of binary classification models. Brier Score is equivalent to the mean squared error, which we calculate for a continuous outcome. A Brier score of 0 indicates perfect accuracy and a score of 1 indicates perfect inaccuracy.

```
require(DescTools)
fit5 <- glm(formula5, family = binomial(), data = analytic3)
BrierScore(fit5)
#> [1] 0.1998676
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

References

R functions (P)

The list of new R functions introduced in this *Predictive factors* lab component are below:

Function_name	Package_name	Use
aggregate	base/stats	To see summary by groups, e.g., by gender
anova	base/stats	To compare models
auc	pROC	To compute the AUC (area under the ROC curve) value
BrierScore	DescTools	To calculate the Brier score
coef	base/stats	To see the coefficients of a fitted model
cor	base/stats	To see the correlation between numeric variables
corrplot	corrplot	To visualize a correlation matrix
createDataPartition	caret	To split a dataset into training and testing sets
createFolds	caret	To create k folds based on the outcome variable
crPlots	car	To see partial residual plot
describeBy	psych	To see summary by groups, e.g., by gender
glm	base/stats	To run generalized linear models
group_by	dplyr	To group by variables
hat	base/stats	To return a hat matrix
ifelse	base	To set an condition, e.g., creating a categorical variable from a numerical
kable	knitr	To create a nice table
layout	base/graphics	To specify plot arrangement
lines	base/graphics	To draw a line graph
lm	base/stats	To fit a linear regression
lowess	base/stats	To smooth a scatter plot
model.matrix	base/stats	To construct a design/model matrix, e.g., a matrix with covariate values
ols_plot_resid_lev	olsrr	To visualize the residuals vs leverage plot
ols_vif_tol	olsrr	To calculate tolerance and variance inflation factor
predict	base/stats	‘predict‘ is a generic function that is used for prediction, e.g., predicting
R2	caret	To calculate the R-squared value
RMSE	caret	To calculate the RMSE value
roc	pROC	To build a ROC curve
sample	base	To take/draw random samples with or without replacement

Function_name	Package_name	Use
save.image	base	To save an R object
spearman2	Hmisc	To compute the square of Spearman's rank correlation
summarize	dplyr	To see summary
tapply	base	To apply a function over an array, e.g., to see the summary of a variable
train	caret	To fit the model with tuning hyperparameters
trainControl	caret	To tune the hyperparameters, i.e., controlling the parameters to train the
varclus	Hmisc	We use the ‘varclus‘ function to identify collinear predictors with cluster
vif	car	To calculate variance inflation factor
which	base	To see which indices are TRUE

Quiz (P)

> ⚠ Warning
>
> We are currently in the beta testing phase for our quiz page updates, so please be aware that some responses may not be accurate. Thank you for your patience.
> For those interested in downloading the quiz questions, you'll find a section below dedicated to downloading.

Live Quiz

You can click on the live quiz options to either see hints of why your response may be incorrect (response marked in red), or whether your choice of response is correct (response marked in green):

Download Quiz

1. **Downloading the File:**
   - Navigate to the link: See here.
   - Right-click on the link and select "Save link as…" from the dropdown menu.
   - Alternatively click here for downloading the quizzes.
   - Choose a destination folder on your computer where you'd like to save the file (e.g., Desktop). **Remember this location**, as you'll need to navigate to it later.
2. **Setting Up RStudio:**

- If you don't have RStudio installed, see the download link in here.
- Launch RStudio after installing.

3. **Installing Necessary R Packages:**

   - Before running the R Markdown file, ensure you have the required packages. In RStudio's console (located at the bottom left by default), enter the following commands to install them:

```
install.packages("learnr")
install.packages("xfun")
```

4. **Opening and Running the File:**

   - In RStudio, go to `File` > `Open File...`.
   - Navigate to the folder where you saved the `Rmd` file and select it to open.
   - Once the file is open in RStudio, you'll see a "Run Document" button (green) at the top of the script editor. Click on it to run the R Markdown Quiz file.

Part VI

Survey data analysis

Background

The chapter consists of a series of tutorials focused on conducting rigorous analyses of complex survey data, mainly using Canadian Community Health Survey (CCHS) and National Health and Nutrition Examination Survey (NHANES) datasets. The tutorials guide users through various stages of survey data analysis: formulating research questions via the PICOT framework, data preparation, quality assessment, and handling missing data. They cover both bivariate and multivariable statistical methods, such as logistic and linear regressions, emphasizing the need to account for complex survey design elements like weights, strata, and clusters to avoid biased estimates. Advanced statistical techniques like backward elimination and interaction effect assessments are also discussed. Predictive model performance is evaluated using metrics like AIC, pseudo R-squared, and ROC curves, along with specialized tests like Archer and Lemeshow Goodness of Fit. The tutorials serve as a comprehensive guide for anyone looking to delve deep into the intricacies of analyzing complex survey data effectively.

> **!** Important
>
> **Datasets**:
> All of the datasets used in this tutorial can be accessed from this GitHub repository folder

The foundation we've built in understanding various research questions, especially the distinction between causal and predictive inquiries, will be instrumental in our next phase. Survey data, with its rich and diverse information, often presents opportunities to address both causal and predictive questions. By leveraging the knowledge we've garnered about the intricacies of causality and the methodologies of prediction, we'll be better equipped to extract meaningful insights from nationally representative survey data. This holistic approach ensures that we not only comprehend the underlying theories but also effectively apply them in real-world contexts, making our analysis robust, relevant, and impactful.

Overview of tutorials

CCHS: Revisiting PICOT

The tutorial focuses on revisiting a research question concerning the relationship between osteoarthritis (OA) and cardiovascular disease (CVD) in Canadian adults, utilizing data from the Canadian Community Health Survey (CCHS) spanning from 2001 to 2005. The approach follows the PICOT framework, which specifies the target population, outcome, exposure and

control groups, and timeline. The tutorial provides detailed steps for data preparation and analysis, from loading the necessary R packages to subsetting data based on a comprehensive set of variables like age, sex, marital status, and income among others. The variables are recoded into broader categories for easier analysis. The tutorial then combines different cycles of CCHS data into one comprehensive dataset. Potential confounders are also identified to better understand the relationship between OA and CVD.

CCHS: Assessing data

This tutorial provides a comprehensive guide to data preparation and quality assessment. It emphasizes the importance of checking for missing data and visualizing it, creating summary tables to look for zero-cells in variables, and generating frequency tables for various variables to examine data distribution. Specific attention is given to the presence of problematic variables. Data dictionaries from different cycles are also consulted to ensure variable compatibility. After identifying and modifying problematic data, the tutorial also explains how to set appropriate reference levels for factors in the dataset and offers an option to create a new dataset that excludes missing values (although this is not generally recommended).

CCHS: Bivariate analysis

This tutorial outlines how to examine relationships between two variables using R. The tutorial covers data preparation steps such as accumulating survey weights across cycles. It also highlights the handling of missing data and survey design specifications for weighted analyses. Descriptive weighted statistics are generated in tables, stratified by exposure and outcome, to provide insights for survey weighted logistic regression analysis. Additionally, proportions and design effects are calculated to account for the complex survey design's impact on statistical estimates. The tutorial employs specialized chi-square tests, such

as, Rao-Scott and Thomas-Rao modifications, to assess associations between variables, accounting for the survey's complex design.

CCHS: Regression analysis

This tutorial offers a comprehensive guide on conducting complex regression analyses on survey data using R. The tutorial starts by conducting basic data checks. It then performs both simple and multivariable logistic regression to explore the relationship between cardiovascular disease and osteoarthritis. Model fit is assessed using Akaike Information Criterion (AIC) and pseudo R-squared metrics. Variable selection techniques such as backward elimination and stepwise regression guided by AIC are applied to hone the model. The tutorial also delves into assessing interaction effects among variables like age, sex, and diabetes, incorporating significant interactions into the final model.

CCHS: Model performance

The tutorial guides users through the process of evaluating logistic regression models fitted to complex survey data in R, focusing primarily on the Receiver Operating Characteristic (ROC) curves and Archer and Lemeshow Goodness of Fit tests. It introduces a specialized function for plotting ROC curves and calculating the Area Under the Curve (AUC) to gauge the model's predictive accuracy, while taking survey weights into account. Grading guidelines for AUC values are provided for model discrimination quality. For model fit, the Archer and Lemeshow test is used. The tutorial also covers additional functionalities for dealing with strata and clusters in the survey data.

NHANES: Predicting blood pressure

The tutorial provides a comprehensive guide for analyzing health survey data with a focus on how demographic factors like race, age, gender, and marital status relate to blood

pressure levels using NHANES dataset. The tutorial constructs both bivariate and multivariate regression models. Additionally, the tutorial incorporates complex survey designs by creating a new survey design object that factors in sampling weight, strata, and clusters. It also generates box plots and summary statistics to visualize variations in blood pressure across different demographic groups, considering survey design. The tutorial emphasizes the importance of accounting for survey design features to avoid biased estimates and discusses the challenges of model overfitting and optimism when shifting from inference to prediction, recommending optimism-correction techniques.

NHANES: Predicting cholesterol level

In the study using NHANES data, the goal was to predict cholesterol levels in adults based on various predictors such as gender, country of birth, race, education, marital status, income, BMI, and diabetes. The data was filtered to include only adults 18 years and older, and multiple statistical tests were conducted. Linear regression and logistic regression models were fitted, with results suggesting an association between gender and cholesterol level. Various statistical tests, including Wald tests and backward elimination, were employed to optimize the model. The study found that income was not a significant predictor for cholesterol levels, and interaction terms did not improve the model. Despite utilizing survey design features, the model had poor discriminatory power. However, Archer-Lemeshow Goodness of Fit test showed that the model fits the data well. The inclusion of age as an additional predictor led to different odds ratios, and the AIC value suggested that adding age improved the model.

NHANES: Properly subsetting a design object

The tutorial provides a comprehensive guide on how to handle and analyze a subset of complex survey data from the NHANES study using R. It begins by checking for missing data. The

focus is on subsetting data based on complete information, emphasizing the importance of accounting for the full complex survey design to obtain unbiased variance estimates. Logistic regression is then run on this subset, with the tutorial explicitly differentiating between correct and incorrect approaches to consider the survey design. Finally, variable selection methods like backward elimination are discussed to determine significant predictors, emphasizing the retention of variables deemed important based on prior research.

Tip

Optional Content:
You'll find that some sections conclude with an optional video walkthrough that demonstrates the code. Keep in mind that the content might have been updated since these videos were recorded. Watching these videos is optional.

Warning

Bug Report:
Fill out this form to report any issues with the tutorial.

Concepts (D)

Survey Data Analysis

Design-based analysis differs from model-based analysis in its approach to handling survey data. Design-based analysis emphasizes the importance of the survey's sampling method and structure, focusing on representativeness and accurate variance estimation according to how the data was collected. It accounts for the complexities of the sampling design, e.g., stratification and clustering, to ensure that results are representative of the entire population. On the other hand, model-based analysis uses statistical models to understand relationships and patterns, assuming data come from a specific distribution and often relying on random sampling.

Understanding survey features such as weights, strata, and clusters is crucial in complex survey data analysis. Survey weights adjust for unequal probabilities of selection and nonresponse, ensuring that the sample represents the population accurately. Stratification improves precision and representation of subgroups, while clustering, often used for practicality and cost considerations, must be accounted for to avoid underestimating standard errors. These features are vital in design-based analysis to provide unbiased, reliable estimates and are what fundamentally distinguish it from model-based approaches, which may not reflect the difficulties of complex survey structures. NHANES is used an an example to explain these ideas.

Reading list

Key reference: (Steven G. Heeringa, West, and Berglund 2017) (chapters 2 and 3)

Optional reading: (Steven G. Heeringa, West, and Berglund 2014)

Theoretical references (optional):

- F/chi-squared statistic with the Rao-Scott second-order correction (Rao and Scott 1984; Koch, Freeman Jr, and Freeman 1975; Thomas and Rao 1987)
- AIC and BIC for modeling with complex survey data (Lumley and Scott 2015)
- Pseudo-R2 statistics under complex sampling (Lumley 2017)
- Tests for regression models fitted to survey data (Lumley and Scott 2014)
- Goodness-of-fit test for a logistic regression model fitted using survey sample data (Archer and Lemeshow 2006)

Video Lessons

Survey Data Analysis

What is included in this Video Lesson:

- reference 00:38
- design-based 1:28
- examples 3:33
- NHANES and sampling 4:54
- weights and other survey features 9:05
- estimate of interest 12:55
- design effect 15:52
- Variance estimation 18:13
- design-based analysis 25:11
- How to make inference 29:33
- inappropriate analysis 32:08
- how useful are sampling weights 36:15
- how useful are psu/cluster info 37:42
- subpopulation / subsetting 38:57
- missingness collected to weights? 40:45
- Dealing with subpopulation 41:38

The timestamps are also included in the YouTube video description.

Video Lesson Slides

Links

- Google Slides
- PDF Slides

References

CCHS: Revisiting PICOT

Welcome to this tutorial where we will examine the same research question presented in the Causal question-1 tutorial. Our approach will be enriched this time by working with a more comprehensive set of covariates. We will follow the guidelines from the research article by Rahman et al. (2013) (DOI:10.1136/bmjopen-2013-002624). We will also will work properly with survey feature variables (e.g., sampling weights).

Rahman et al. (2013)

Remembering PICOT

Before diving into the data, let's clarify and remember the research parameters using the PICOT framework:

(1) **Target population**: Canadian adults (CCHS data)

(2) **Outcome** (Y): CVD (Heart disease/Cardiovascular Disease)

(3) **Exposure group** (A): Osteoarthritis (OA)

(4) **Control group**: People without OA

(5) **Timeline**: Data collected from 2001 to 2005

In addition, we'll identify potential confounders based on literature to better understand the relationship between exposure and outcome.

Creating dataset

To start, we'll load the required R packages:

```
# Load required packages
library(survey)
library(knitr)
library(car)
```

Load data

We'll be using CCHS data from various cycles. Use the following code to load this data into your R environment:

```
# CCHS 1.1
load("Data/surveydata/cchsc1.RData")

# CCHS 2.1
load("Data/surveydata/cchsc2.RData")

# CCHS 3.1
load("Data/surveydata/cchsc3.RData")

# objects loaded
ls()
#> [1] "c1"                "c2"                "c3"                "has_annotations"
```

To check the dimensions of each data set:

```
# Dimensions
dim(c1)
#> [1] 130880    117
dim(c2)
#> [1] 134072    112
dim(c3)
#> [1] 132221    112
```

Subset the data (subset variables)

Understand variable coding

Before subsetting the data, we need to comprehend the variables used in CCHS 3.1, which are described in detail in the official reference:

- page 60 in CCHS 3.1 guide (Canada 2005)
- link for variable list: (E. Karim 2023)

A table mapping variable concepts across different CCHS cycles can be found below.

Variable Concept	CCHS 1.1	CCHS 2.1	CCHS 3.1
Has heart disease	CCCA_121	CCCC_121	CCCE_121
Has arthritis or rheumatism	CCCA_051	CCCC_051	CCCE_051
Kind of arthritis	CCCA_05A	CCCC_05A	CCCE_05A
Age	DHHAGAGE	DHHCGAGE	DHHEGAGE
Sex	DHHA_SEX	DHHC_SEX	DHHE_SEX
Marital Status	DHHAGMS	DHHCGMS	DHHEGMS
Cultural / racial origin	SDCAGRAC	SDCCGRAC	SDCEGCGT
Immigrant status	SDCAFIMM	SDCCFIMM	SDCEFIMM
Length of time in Canada since immigration	SDCAGRES	SDCCGRES	SDCEGRES
Highest level of education - respondent	EDUADR04	EDUCDR04	EDUEDR04

Variable Concept	CCHS 1.1	CCHS 2.1	CCHS 3.1
Total household income from all sources	INCAGHH	INCCGHH	INCEGHH
Body mass index	HWTAGBMI	HWTCGBMI	HWTEGBMI
Physical activity index	PACADPAI	PACCDPAI	PACEDPAI
Has a regular medical doctor	TWDA_5	HCUC_1AA	HCUE_1AA
Self-perceived stress	GENA_07	GENC_07	GENE_07
Type of smoker	SMKADSTY	SMKCDSTY	SMKEDSTY
Type of drinker	ALCADTYP	ALCCDTYP	ALCEDTYP
Daily consumption - total fruits and vegetables	FVCADTOT	FVCCDTOT	FVCEDTOT
Has high blood pressure	CCCA_071	CCCC_071	CCCE_071
Has emphysema or chronic obstructive pulmonary disease (COPD)	CCCA_91B	CCCC_91B	CCCE_91F
Has diabetes	CCCA_101	CCCC_101	CCCE_101
Province	GEOAGPRV	GEOCGPRV	GEOEGPRV
Sampling weight - master weight	WTSAM	WTSC_M	WTSE_M

While most variables in CCHS 3.1 are universally applicable to 'All respondents,' there are some exceptions. For example:

- CCCE_05A universe: (Kind of arthritis / rheumatism)
- Respondents who answered CCCE_051 = (1, 7 or 8) or CCCE_011 = 8
- CCCE_051: All respondents
- CCCE_011: All respondents
- SDCEGRES universe: (Length of time in Canada since immigration)
  - Respondents who answered SDCE_2 = (2, 7 or 8) or SDCE_1 = (97 or 98)
  - SDCE_2 doesn't exist!
  - (master file variable; not available in PUMF)
  - Public Use Microdata File (PUMF)
  - SDCE_1 doesn't exist!
  - (master file variable; not available in PUMF)
- HWTEGBMI universe: (Body Mass Index (BMI) / self-report)
  - All respondents excluding pregnant women (MAME_037 = 1)
    - MAME_037 doesn't exist!
    - (master file variable; not available in PUMF)
- GENE_07 universe: (Self-perceived stress)
  - Respondents aged 15 and over
- CCCE_91F universe: (Has chronic obstructive pulmonary disease)
  - Respondents aged 30 and over
- FVCEDTOT universe: (Daily consumption - total fruits and vegetables)
  - Respondents with FVCEFOPT = 1

    * FVCEFOPT: Optional module: Fruit and vegetable consumption - (F)

Ref:

- page 66 in CCHS 3.1 guide (Canada 2005)
- Potential problematic variables:
  - Self-perceived stress
  - Has chronic obstructive pulmonary disease / copd
  - Daily consumption - total fruits and vegetables

We will make decisions about these variables later: for now, let's keep them.

Restrict the dataset with variables of interest only

0.0.0.0.1 * Cycle 1.1

- We define a vector of variable names `var.names1` that are of interest for the first cycle of the Canadian Community Health Survey (CCHS 1.1). These variables cover a range of topics such as heart disease, age, sex, etc.
- Then we creates a new data frame `cc11` by subsetting the original data frame `c1` to include only the columns specified in `var.names1`.

```
var.names1 <- c("CCCA_121", "CCCA_051", "CCCA_05A", "DHHAGAGE",
                "DHHA_SEX", "DHHAGMS", "SDCAGRAC", "SDCAFIMM",
                "SDCAGRES", "EDUADR04", "INCAGHH", "HWTAGBMI",
                "PACADPAI", "TWDA_5", "GENA_07", "SMKADSTY",
                "ALCADTYP", "FVCADTOT", "CCCA_071", "CCCA_91B",
                "CCCA_101", "GEOAGPRV", "WTSAM")
cc11 <- c1[var.names1]
dim(cc11)
#> [1] 130880     23
table(cc11$CCCA_051)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          24511         106231              0            110              3
```

```
#>     NOT STATED
#>             25
```

The subsequent two code chunks do the same for CCHS 2.1 and CCHS 3.1, respectively, resulting in new data frames `cc21` and `cc31`.

0.0.0.0.2 * Cycle 2.1

```
var.names2 <- c("CCCC_121", "CCCC_051", "CCCC_05A", "DHHCGAGE",
                "DHHC_SEX", "DHHCGMS", "SDCCGRAC", "SDCCFIMM",
                "SDCCGRES", "EDUCDR04", "INCCGHH", "HWTCGBMI",
                "PACCDPAI", "HCUC_1AA", "GENC_07", "SMKCDSTY",
                "ALCCDTYP", "FVCCDTOT", "CCCC_071", "CCCC_91B",
                "CCCC_101", "GEOCGPRV", "WTSC_M")
cc21 <- c2[var.names2]
dim(cc21)
#> [1] 134072     23
table(cc21$CCCC_051)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          29293         104530              0            208             11
#>     NOT STATED
#>             30
```

0.0.0.0.3 * Cycle 3.1

```
var.names3 <- c("CCCE_121", "CCCE_051", "CCCE_05A", "DHHEGAGE",
                "DHHE_SEX", "DHHEGMS", "SDCEGCGT", "SDCEFIMM",
                "SDCEGRES", "EDUEDR04", "INCEGHH", "HWTEGBMI",
                "PACEDPAI", "HCUE_1AA", "GENE_07", "SMKEDSTY",
                "ALCEDTYP", "FVCEDTOT","CCCE_071", "CCCE_91F",
                "CCCE_101", "GEOEGPRV", "WTSE_M")
cc31 <- c3[var.names3]
dim(cc31)
#> [1] 132221     23
table(cc31$CCCE_051)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
```

```
#>          28221          103781               0          191                4
#>     NOT STATED
#>             24
```

Making variable names the same

We now create a new set of more readable and consistent variable names.

```
new.var.names <- c("CVD", "arthritis", "arthritis.kind", "age",
                   "sex", "married", "race", "immigration",
                   "recent.immigrant", "edu", "income", "bmi",
                   "phyact", "doctor", "stress", "smoke",
                   "drink", "fruit", "bp", "copd",
                   "diab",  "province", "weight")
cbind(new.var.names, var.names1, var.names2, var.names3)
#>       new.var.names      var.names1 var.names2 var.names3
#>  [1,] "CVD"              "CCCA_121" "CCCC_121" "CCCE_121"
#>  [2,] "arthritis"        "CCCA_051" "CCCC_051" "CCCE_051"
#>  [3,] "arthritis.kind"   "CCCA_05A" "CCCC_05A" "CCCE_05A"
#>  [4,] "age"              "DHHAGAGE" "DHHCGAGE" "DHHEGAGE"
#>  [5,] "sex"              "DHHA_SEX" "DHHC_SEX" "DHHE_SEX"
#>  [6,] "married"          "DHHAGMS"  "DHHCGMS"  "DHHEGMS"
#>  [7,] "race"             "SDCAGRAC" "SDCCGRAC" "SDCEGCGT"
#>  [8,] "immigration"      "SDCAFIMM" "SDCCFIMM" "SDCEFIMM"
#>  [9,] "recent.immigrant" "SDCAGRES" "SDCCGRES" "SDCEGRES"
#> [10,] "edu"              "EDUADR04" "EDUCDR04" "EDUEDR04"
#> [11,] "income"           "INCAGHH"  "INCCGHH"  "INCEGHH"
#> [12,] "bmi"              "HWTAGBMI" "HWTCGBMI" "HWTEGBMI"
#> [13,] "phyact"           "PACADPAI" "PACCDPAI" "PACEDPAI"
#> [14,] "doctor"           "TWDA_5"   "HCUC_1AA" "HCUE_1AA"
#> [15,] "stress"           "GENA_07"  "GENC_07"  "GENE_07"
#> [16,] "smoke"            "SMKADSTY" "SMKCDSTY" "SMKEDSTY"
#> [17,] "drink"            "ALCADTYP" "ALCCDTYP" "ALCEDTYP"
#> [18,] "fruit"            "FVCADTOT" "FVCCDTOT" "FVCEDTOT"
#> [19,] "bp"               "CCCA_071" "CCCC_071" "CCCE_071"
#> [20,] "copd"             "CCCA_91B" "CCCC_91B" "CCCE_91F"
#> [21,] "diab"             "CCCA_101" "CCCC_101" "CCCE_101"
#> [22,] "province"         "GEOAGPRV" "GEOCGPRV" "GEOEGPRV"
```

```
#> [23,] "weight"           "WTSAM"    "WTSC_M"   "WTSE_M"
names(cc11) <- names(cc21) <- names(cc31) <- new.var.names

table(cc11$arthritis)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          24511         106231              0            110              3
#>     NOT STATED
#>             25
table(cc21$arthritis)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          29293         104530              0            208             11
#>     NOT STATED
#>             30
table(cc31$arthritis)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          28221         103781              0            191              4
#>     NOT STATED
#>             24

cc11$cycle <- 11
cc21$cycle <- 21
cc31$cycle <- 31
```

Appending

We now combine the data frames `cc11`, `cc21`, and `cc31` by stacking them on top of each other.

```
cc123a <- rbind(cc11,cc21,cc31)
dim(cc123a)
#> [1] 397173     24
names(cc123a)
#>  [1] "CVD"              "arthritis"        "arthritis.kind"   "age"
#>  [5] "sex"              "married"          "race"             "immigration"
#>  [9] "recent.immigrant" "edu"              "income"           "bmi"
#> [13] "phyact"           "doctor"           "stress"           "smoke"
```

```
#> [17] "drink"            "fruit"            "bp"               "copd"
#> [21] "diab"             "province"         "weight"           "cycle"
cc123a$ID <- 1:nrow(cc123a)
```

Variables

Sampling weight

We use the `summary` function to provide basic statistics (like mean, median, min, max, etc.) for the `weight` column in the data frame `cc123a`.

```
summary(cc123a$weight)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    1.17   65.28  126.63  200.09  243.21 7154.95
```

Exposure

This following chunk creates frequency tables for the 'arthritis' and 'arthritis.kind' columns.

```
table(cc123a$arthritis)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          82025         314542              0            509             18
#>     NOT STATED
#>             79
table(cc123a$arthritis.kind)
#>
#> RHEUMATOID ARTH  OSTEOARTHRITIS           OTHER  NOT APPLICABLE      DON'T KNOW
#>           19099           40943            7305          314542           12354
#>         REFUSAL      NOT STATED      RHEUMATISM
#>             215             619            2096
sum(cc123a$arthritis=="NO")
#> [1] 314542
sum(cc123a$arthritis.kind=="NOT APPLICABLE")
#> [1] 314542
```

We create the exposure variable with exposure status vs controls.

```
c123sub1 <- cc123a
c123sub1$arthritis.kind <- car::recode(c123sub1$arthritis.kind,
                            "'OSTEOARTHRITIS'='OSTEOARTHRITIS';
                            'NOT APPLICABLE' = 'NOT APPLICABLE';
                            else = NA",
                            as.factor = FALSE)
table(c123sub1$arthritis.kind, useNA = "always")
#>
#> NOT APPLICABLE OSTEOARTHRITIS           <NA>
#>         314542          40943          41688
```

```
# c123sub1 <- subset(cc123a, arthritis.kind == "OSTEOARTHRITIS" |
#                       arthritis.kind == "NOT APPLICABLE" )
# dim(c123sub1)
table(c123sub1$arthritis.kind)
#>
#> NOT APPLICABLE OSTEOARTHRITIS
#>         314542          40943
table(c123sub1$arthritis)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW         REFUSAL
#>          82025         314542              0            509              18
#>     NOT STATED
#>             79
require(car)
c123sub1$arthritis.kind <- car::recode(c123sub1$arthritis.kind,
                                  "'OSTEOARTHRITIS'='OA';
                         'NOT APPLICABLE'='Control';
                         else=NA", as.factor = FALSE)
table(c123sub1$arthritis.kind, useNA = "always")
#>
#> Control      OA    <NA>
#>  314542   40943   41688
c123sub1$OA <- c123sub1$arthritis.kind
c123sub1$arthritis.kind <- NULL
c123sub1$arthritis <- NULL
dim(c123sub1)
#> [1] 397173     24
```

```r
dim(c123sub1)[1]-dim(cc123a)[1]
#> [1] 0
```

Outcome

We create the outcome variable.

```r
c123sub2 <- c123sub1
table(c123sub2$CVD)
#>
#>             YES              NO NOT APPLICABLE     DON'T KNOW         REFUSAL
#>           25524          371121               0            399              50
#>     NOT STATED
#>              79
c123sub2$CVD <- car::recode(c123sub2$CVD,
                            "'YES'='YES';
                            'NO' = 'NO';
                            else = NA",
                            as.factor = FALSE)
```

```r
# table(c123sub1$CVD)
# c123sub2 <- subset(c123sub1, CVD == "YES" | CVD == "NO")
# table(c123sub2$CVD)
dim(c123sub2)
#> [1] 397173     24
c123sub2$CVD <- car::recode(c123sub2$CVD,
                       "'YES'='event';
                       'NO'='no event';
                       else=NA",
                       as.factor = FALSE)
table(c123sub2$CVD, useNA = "always")
#>
#>    event no event     <NA>
#>    25524   371121      528
dim(c123sub2)
#> [1] 397173     24
dim(c123sub2)[1]-dim(c123sub1)[1]
#> [1] 0
```

Covariates

0.0.0.0.1 * Age

Recodes the 'age' column into broader age categories.

```
c123sub2$age <- car::recode(c123sub2$age,
                            "c('12 TO 14 YEARS','15 TO 19 YEARS',
                            '15 TO 17 YEARS', '18 TO 19 YEARS')='teen';
                            c('20 TO 24 YEARS','25 TO 29 YEARS')=
                            '20-29 years';
                            c('30 TO 34 YEARS','35 TO 39 YEARS')=
                            '30-39 years';
                            c('40 TO 44 YEARS','45 TO 49 YEARS')=
                            '40-49 years';
                            c('50 TO 54 YEARS','55 TO 59 YEARS')=
                            '50-59 years';
                            c('60 TO 64 YEARS')='60-64 years';
                            else='65 years and over'",
                            as.factor = FALSE)
table(c123sub2$age)
#>
#>       20-29 years       30-39 years       40-49 years       50-59 years
#>             48652             63810             65111             61035
#>       60-64 years 65 years and over              teen
#>             25265             80913             52387
```

```
c123sub3 <- c123sub2
# c123sub3$age[c123sub3$age == 'teen'] <- NA
# c123sub3$age[c123sub3$age == '65 years and over'] <- NA
```

```
# dim(c123sub2)
# c123sub3 <- subset(c123sub2, age != 'teen' & age != '65 years and over')
table(c123sub3$age, useNA = "always")
#>
#>       20-29 years       30-39 years       40-49 years       50-59 years
#>             48652             63810             65111             61035
#>       60-64 years 65 years and over              teen              <NA>
#>             25265             80913             52387                 0
dim(c123sub3)
#> [1] 397173     24
```

0.0.0.0.2 * Sex

Recodes 'sex' to 'Male' or 'Female'.

```
table(c123sub3$sex)
#>
#>           MALE         FEMALE NOT APPLICABLE     NOT STATED     DON'T KNOW
#>         182523         214650              0              0              0
#>        REFUSAL
#>              0
c123sub3$sex <- car::recode(c123sub3$sex,
                            "'MALE'='Male';
                            'FEMALE' = 'Female';
                            else = NA",
                            as.factor = FALSE)
table(c123sub3$sex, useNA = "always")
#>
#> Female   Male   <NA>
#> 214650 182523      0
```

0.0.0.0.3 * Marital status

Recodes the 'married' column into two categories: 'not single' and 'single'.

```
table(c123sub3$married)
#>
#>          MARRIED       COMMON-LAW    WIDOW/SEP/DIV           SINGLE
#>           171304            30909            75885            78261
#>          REFUSAL       NOT STATED   NOT APPLICABLE       DON'T KNOW
#>                0              661                0                0
#> SINGLE/NEVER MAR
#>            40153
c123sub3$married <- car::recode(c123sub3$married,
                            "c('MARRIED', 'COMMON-LAW')='not single';
                            c('WIDOW/SEP/DIV', 'SINGLE',
                            'SINGLE/NEVER MAR') = 'single';
                            else = NA",
                            as.factor = FALSE)
table(c123sub3$married, useNA = "always")
#>
```

```
#> not single     single       <NA>
#>     202213     194299        661
```

0.0.0.0.4 * Race/ethnicity

Recodes 'race' into 'White' and 'Non-white'.

```
table(c123sub3$race)
#>
#>            WHITE VISIBLE MINORITY   NOT APPLICABLE       NOT STATED
#>           349222            38641                0             9310
#>       DON'T KNOW          REFUSAL
#>                0                0
c123sub3$race <- car::recode(c123sub3$race,
                             "'WHITE'='White';
                             'VISIBLE MINORITY' = 'Non-white';
                             else = NA",
                             as.factor = FALSE)
table(c123sub3$race, useNA = "always")
#>
#> Non-white     White      <NA>
#>     38641    349222      9310
```

0.0.0.0.5 * Immigration

Creates a new column for immigration status based on the 'recent.immigrant' column, then removes the original column.

```
table(c123sub3$recent.immigrant)
#>
#>     0 TO 9 YEARS 10 YEARS OR MORE   NOT APPLICABLE       NOT STATED
#>            10644            26746           338078             7975
#>       DON'T KNOW          REFUSAL 10 OR MORE YEARS
#>                0                0            13730
c123sub3$immigrate <- car::recode(c123sub3$recent.immigrant,
                            "'0 TO 9 YEARS'='recent';
                            '10 YEARS OR MORE' = '> 10 years';
                            'NOT APPLICABLE' = 'not immigrant';
                            else = NA",
```

```r
                              as.factor = FALSE)
table(c123sub3$immigrate, useNA = "always")
#>
#>    > 10 years not immigrant        recent          <NA>
#>         26746        338078         10644         21705
c123sub3$recent.immigrant <- NULL
c123sub3$immigration <- NULL
```

0.0.0.0.6 * Education

Recode educational status into specified categories.

```r
table(c123sub3$edu)
#>
#> < THAN SECONDARY  SECONDARY GRAD.  OTHER POST-SEC.  POST-SEC. GRAD.
#>           124425            64753            29000           171972
#>   NOT APPLICABLE       NOT STATED       DON'T KNOW          REFUSAL
#>                0             7023                0                0
c123sub3$edu <- car::recode(c123sub3$edu,
                              "'< THAN SECONDARY'='< 2ndary';
                              'SECONDARY GRAD.' = '2nd grad.';
                              'POST-SEC. GRAD.' = 'Post-2nd grad.';
                              'OTHER POST-SEC.' = 'Other 2nd grad.';
                              else = NA",
                              as.factor = FALSE)
table(c123sub3$edu, useNA = "always")
#>
#>        < 2ndary       2nd grad. Other 2nd grad.  Post-2nd grad.            <NA>
#>          124425           64753           29000         171972            7023
```

0.0.0.0.7 * Income

Recodes income levels into broader categories.

```r
table(c123sub3$income)
#>
#>        NO INCOME LESS THAN 15,000  $15,000-$29,999  $30,000-$49,999
#>            12636            14103            63766            77940
#>  $50,000-$79,999  $80,000 OR MORE   NOT APPLICABLE       NOT STATED
```

```
#>            85108            75715                0            57079
#>       DON'T KNOW          REFUSAL   NO OR <$15,000
#>                0                0            10826
# cycle 1.1 has: 'NO INCOME','LESS THAN 15,000'
# Other cycles have: 'NO OR <$15,000'
c123sub3$income <- car::recode(c123sub3$income,
                               "c('NO OR <$15,000', 'NO INCOME',
                               'LESS THAN 15,000',
                               '$15,000-$29,999')='$29,999 or less';
                               '$30,000-$49,999' = '$30,000-$49,999';
                               '$50,000-$79,999' = '$50,000-$79,999';
                               '$80,000 OR MORE' = '$80,000 or more';
                               else = NA",
                               as.factor = FALSE)
table(c123sub3$income, useNA = "always")
#>
#> $29,999 or less $30,000-$49,999 $50,000-$79,999 $80,000 or more            <NA>
#>          101331           77940           85108           75715           57079
```

0.0.0.0.8 * BMI

Converts 'bmi' column values to numerical data and also categorizes it based on the BMI value.

If you want to reuse the continuous variable later (usually a good idea in statistical sense), keep a second copy of the variable.

```
# table(c123sub3$bmi)
sum(c123sub3$bmi=="NOT APPLICABLE")+
  sum(c123sub3$bmi=="REFUSAL")+
  sum(c123sub3$bmi=="NOT STATED")+
  sum(c123sub3$bmi=="DON'T KNOW")
#> [1] 72303
c123sub3$bmi <- car::recode(c123sub3$bmi,
                               'c("NOT APPLICABLE", "REFUSAL",
                               "NOT STATED", "DON\'T KNOW")=NA',
                               as.factor = FALSE)
#table(c123sub3$bmi, useNA = "always")
c123sub3$bmi <- as.numeric(as.character(c123sub3$bmi))
```

```r
summary(c123sub3$bmi)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>   11.91   22.47   25.30   25.86   28.50   57.90   72303
c123sub3$bmi2 <- cut(c123sub3$bmi,
                     breaks = c(0,18.5,25,Inf),
                     right = TRUE,
                     labels = c("Underweight",
                                "healthy weight",
                                "Overweight"))
c123sub3$bmi <- c123sub3$bmi2
c123sub3$bmi2 <- NULL
table(c123sub3$bmi, useNA = "always")
#>
#>    Underweight healthy weight     Overweight           <NA>
#>          10116         148533         166221          72303
```

0.0.0.0.9 * Physical activity

Recodes physical activity levels.

```r
table(c123sub3$phyact)
#>
#>         ACTIVE       MODERATE       INACTIVE NOT APPLICABLE     NOT STATED
#>          98571          93507         190739              0          14356
#>     DON'T KNOW        REFUSAL
#>              0              0
c123sub3$phyact <- car::recode(c123sub3$phyact,
                               "'ACTIVE'='Active';
                               'MODERATE' = 'Moderate';
                               'INACTIVE' = 'Inactive';
                               else = NA",
                               as.factor = FALSE)
table(c123sub3$phyact, useNA = "always")
#>
#>   Active Inactive Moderate     <NA>
#>    98571   190739    93507    14356
```

0.0.0.0.10 * Doctor

Recodes whether someone has access to a doctor into 'Yes' or 'No'.

```
table(c123sub3$doctor)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW         REFUSAL
#>         338105          58623              0            406              39
#>     NOT STATED
#>              0
c123sub3$doctor <- car::recode(c123sub3$doctor,
                            "'YES'='Yes';
                            'NO' = 'No';
                            else = NA",
                            as.factor = FALSE)
table(c123sub3$doctor, useNA = "always")
#>
#>     No    Yes   <NA>
#>  58623 338105    445
```

0.0.0.0.11 * Stress

Recodes stress levels into 'Not too stressed' or 'stressed'.

```
table(c123sub3$stress)
#>
#>     NOT AT ALL       NOT VERY          A BIT    QUITE A BIT      EXTREMELY
#>          51020          89612         146112          68203          14094
#> NOT APPLICABLE     DON'T KNOW        REFUSAL     NOT STATED
#>          26737           1244            145              6
c123sub3$stress <- car::recode(c123sub3$stress,
                              "c('NOT AT ALL','NOT VERY','A BIT')=
                              'Not too stressed';
                              c('QUITE A BIT','EXTREMELY') =
                              'stressed';
                              else = NA",
                              as.factor = FALSE)
table(c123sub3$stress, useNA = "always")
#>
#> Not too stressed         stressed             <NA>
#>           286744            82297            28132
```

0.0.0.0.12 * Smoking

Recodes smoking status into 'Current smoker', 'Former smoker', or 'Never smoker'.

```
table(c123sub3$smoke)
#>
#>             DAILY        OCCASIONAL ALWAYS OCCASION.      FORMER DAILY
#>             79878             10931             7043             99450
#> FORMER OCCASION.      NEVER SMOKED   NOT APPLICABLE        NOT STATED
#>             58120            140017                0              1734
#>        DON'T KNOW           REFUSAL
#>                 0                 0
c123sub3$smoke <- car::recode(c123sub3$smoke,
                              "c('DAILY','OCCASIONAL',
                              'ALWAYS OCCASION.')='Current smoker';
                              c('FORMER DAILY','FORMER OCCASION.',
                              'ALWAYS OCCASION.') = 'Former smoker';
                              'NEVER SMOKED' = 'Never smoker';
                              else = NA",
                              as.factor = FALSE)
table(c123sub3$smoke, useNA = "always")
#>
#> Current smoker  Former smoker   Never smoker           <NA>
#>          97852         157570         140017           1734
```

0.0.0.0.13 * Alcohol

Recodes drinking habits into 'Current drinker', 'Former driker', or 'Never drank'.

```
table(c123sub3$drink)
#>
#> REGULAR DRINKER    OCC. DRINKER  FORMER DRINKER     NEVER DRANK  NOT APPLICABLE
#>          219334           76399           55299           40670               0
#>      NOT STATED      DON'T KNOW         REFUSAL
#>            5471               0               0
c123sub3$drink <- car::recode(c123sub3$drink,
                              "c('REGULAR DRINKER',
                              'OCC. DRINKER')='Current drinker';
                              c('FORMER DRINKER') = 'Former driker';
```

```
                                 'NEVER DRANK' = 'Never drank';
                                 else = NA",
                                 as.factor = FALSE)
table(c123sub3$drink, useNA = "always")
#>
#> Current drinker   Former driker     Never drank            <NA>
#>          295733           55299           40670            5471
```

0.0.0.0.14 * Fruit and vegetable consumption

Converts fruit and vegetable consumption to numerical values and then categorizes it.

If you want to reuse the continuous variable later (usually a good idea in statistical sense), keep a second copy of the variable.

```
str(c123sub3$fruit)
#>  Factor w/ 303 levels "0","0.1","0.2",..: 57 48 42 72 67 120 39 54 61 51 ...
#c123sub3$fruit.cont <- c123sub3$fruit
c123sub3$fruit2 <- as.numeric(as.character(c123sub3$fruit))
# Note: do not use as.numeric(c123sub3$fruit)
summary(c123sub3$fruit2)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>    0.00    3.00    4.30    4.75    6.00   80.00   75347
c123sub3$fruit2 <- cut(c123sub3$fruit2,
                       breaks = c(0,3,6,Inf),
                       right = TRUE,
                       labels = c("0-3 daily serving",
                                  "4-6 daily serving",
                                  "6+ daily serving"))
table(c123sub3$fruit2, useNA = "always")
#>
#> 0-3 daily serving 4-6 daily serving  6+ daily serving              <NA>
#>             83441            159380             78857             75495
c123sub3$fruit <- c123sub3$fruit2
c123sub3$fruit2 <- NULL
```

0.0.0.0.15 * Hypertension

The following chunk of code is concerned with recoding the blood pressure column. The code also shows the distribution of the values before and after recoding.

```
table(c123sub3$bp)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW         REFUSAL
#>          68071         328164              0            793              66
#>     NOT STATED
#>             79
c123sub3$bp <- car::recode(c123sub3$bp,
                           "'YES'='Yes';
                           'NO' = 'No';
                           else = NA",
                           as.factor = FALSE)
table(c123sub3$bp, useNA = "always")
#>
#>     No    Yes   <NA>
#> 328164  68071    938
```

0.0.0.0.16 * COPD

The following chunk of code is concerned with recoding the COPD column.

```
table(c123sub3$copd)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW         REFUSAL
#>           4508         291191         101039            329              26
#>     NOT STATED
#>             80
c123sub3$copd <- car::recode(c123sub3$copd,
                             "'YES'='Yes';
                             'NO' = 'No';
                             else = NA",
                             as.factor = FALSE)
table(c123sub3$copd, useNA = "always")
#>
#>     No    Yes   <NA>
#> 291191   4508 101474
```

0.0.0.0.17 * Diabetes

The following chunk of code is concerned with recoding the diabetes column.

```
table(c123sub3$diab)
#>
#>            YES             NO NOT APPLICABLE     DON'T KNOW        REFUSAL
#>          22231         374589              0            236             38
#>     NOT STATED
#>             79
c123sub3$diab <- car::recode(c123sub3$diab,
                             "'YES'='Yes';
                             'NO' = 'No';
                             else = NA",
                             as.factor = FALSE)
table(c123sub3$diab, useNA = "always")
#>
#>     No    Yes   <NA>
#> 374589  22231    353
```

0.0.0.0.18 * North

This section is concerned with recoding the province column. It groups provinces into "North" and "South" based on their names.

- Note that the category names might not exactly match with the data dictionary:
  - Particularly note two versions of `QUEBEC`
  - `PEI` was written in short
  - None in
    - DON'T KNOW
    - REFUSAL
    - NOT STATED
    - NOT APPLICABLE

```
c123sub3$province.check <- c123sub3$province
table(c123sub3$province)
#>
```

```
#>     NEWFOUNDLAND              PEI      NOVA SCOTIA    NEW BRUNSWICK
#>             7924             7744            15341            15025
#>        QU\xc9BEC          ONTARIO         MANITOBA     SASKATCHEWAN
#>            22012           123821            23454            23361
#>          ALBERTA BRITISH COLUMBIA YUKON/NWT/NUNAVT   NOT APPLICABLE
#>            40127            49767             5064                0
#>       DON'T KNOW          REFUSAL       NOT STATED           QUEBEC
#>                0                0                0            56764
#>      NFLD & LAB.  YUKON/NWT/NUNA.
#>             4111             2658
c123sub3$province <- car::recode(c123sub3$province,
                                 "c('YUKON/NWT/NUNAVT','YUKON/NWT/NUNA.')=
                                 'North'; else = 'South'",
                                 as.factor = FALSE)
table(c123sub3$province, useNA = "always")
#>
#>  North  South   <NA>
#>   7722 389451      0
```

0.0.0.0.19 * Dimension testing

This chunk verifies the dimensions of the modified data frames and performs some other operations like setting factors.

```
save.c123sub3 <- c123sub3
names(c123sub3)
#>  [1] "CVD"            "age"            "sex"            "married"
#>  [5] "race"           "edu"            "income"         "bmi"
#>  [9] "phyact"         "doctor"         "stress"         "smoke"
#> [13] "drink"          "fruit"          "bp"             "copd"
#> [17] "diab"           "province"       "weight"         "cycle"
#> [21] "ID"             "OA"             "immigrate"      "province.check"
# c123sub3$phyact <- NULL
# c123sub3$stress <- NULL
# c123sub3$fruit <- NULL
# c123sub3$copd <- NULL
# c123sub3$province.check <- NULL
dim(c123sub3)
#> [1] 397173     24
dim(c123sub2)[1]-dim(c123sub3)[1]
```

```r
#> [1] 0
analytic <- c123sub3
dim(analytic)
#> [1] 397173     24
analytic$cycle <- as.factor(analytic$cycle)
names(analytic)
#>  [1] "CVD"            "age"            "sex"            "married"
#>  [5] "race"           "edu"            "income"         "bmi"
#>  [9] "phyact"         "doctor"         "stress"         "smoke"
#> [13] "drink"          "fruit"          "bp"             "copd"
#> [17] "diab"           "province"       "weight"         "cycle"
#> [21] "ID"             "OA"             "immigrate"      "province.check"
```

Saving data

We save the data for future use:

```r
save(cc123a, analytic, file = "Data/surveydata/cchs123.RData")
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

References

CCHS: Assessing data

Let us load all the necessary packages for data manipulation, statistical analysis, and plotting.

```
# Load required packages
library(survey)
library(knitr)
library(car)
library(tableone)
library(DataExplorer)
```

Load data

Data loading that we saved earlier:

```
load("Data/surveydata/cchs123.RData")
ls()
#> [1] "analytic"        "cc123a"          "has_annotations"
```

Checking

Check the data for missingness

Checks the dimensions of the data and runs functions to explore missing data, stratifying by some variables. Additionally, it plots the missing data for visualization.

```
dim(analytic)
#> [1] 397173     24
require("tableone")
#CreateTableOne(data = analytic, includeNA = TRUE)
CreateTableOne(data = analytic, strata = "CVD", includeNA = TRUE)
```

```
#>                        Stratified by CVD
#>                         event                 no event              p      test
#>   n                        25524                371121
#>   CVD (%)                                                                   NaN
#>      event                 25524 (100.0)             0 (  0.0)
#>      no event                  0 (  0.0)        371121 (100.0)
#>      NA                        0 (  0.0)             0 (  0.0)
#>   age (%)                                                             <0.001
#>      20-29 years             330 (  1.3)         48293 ( 13.0)
#>      30-39 years             580 (  2.3)         63194 ( 17.0)
#>      40-49 years            1498 (  5.9)         63549 ( 17.1)
#>      50-59 years            3635 ( 14.2)         57300 ( 15.4)
#>      60-64 years            2720 ( 10.7)         22497 (  6.1)
#>      65 years and over     16496 ( 64.6)         64198 ( 17.3)
#>      teen                    265 (  1.0)         52090 ( 14.0)
#>   sex = Male (%)           12506 ( 49.0)        169776 ( 45.7)     <0.001
#>   married (%)                                                         <0.001
#>      not single            13287 ( 52.1)        188687 ( 50.8)
#>      single                12207 ( 47.8)        181811 ( 49.0)
#>      NA                       30 (  0.1)           623 (  0.2)
#>   race (%)                                                            <0.001
#>      Non-white              1276 (  5.0)         37323 ( 10.1)
#>      White                 23629 ( 92.6)        325178 ( 87.6)
#>      NA                      619 (  2.4)          8620 (  2.3)
#>   edu (%)                                                             <0.001
#>      < 2ndary              11547 ( 45.2)        112678 ( 30.4)
#>      2nd grad.              3310 ( 13.0)         61355 ( 16.5)
#>      Other 2nd grad.        1323 (  5.2)         27643 (  7.4)
#>      Post-2nd grad.         8744 ( 34.3)        163052 ( 43.9)
#>      NA                      600 (  2.4)          6393 (  1.7)
#>   income (%)                                                          <0.001
#>      $29,999 or less       11664 ( 45.7)         89506 ( 24.1)
#>      $30,000-$49,999        4871 ( 19.1)         72994 ( 19.7)
#>      $50,000-$79,999        3193 ( 12.5)         81861 ( 22.1)
#>      $80,000 or more        1905 (  7.5)         73768 ( 19.9)
#>      NA                     3891 ( 15.2)         52992 ( 14.3)
#>   bmi (%)                                                             <0.001
#>      Underweight             504 (  2.0)          9600 (  2.6)
#>      healthy weight         7176 ( 28.1)        141200 ( 38.0)
#>      Overweight            12104 ( 47.4)        153887 ( 41.5)
```

```
#>      NA                    5740 ( 22.5)        66434 ( 17.9)
#>   phyact (%)                                                     <0.001
#>      Active                3642 ( 14.3)        94844 ( 25.6)
#>      Inactive             15494 ( 60.7)       174976 ( 47.1)
#>      Moderate              4928 ( 19.3)        88480 ( 23.8)
#>      NA                    1460 (  5.7)        12821 (  3.5)
#>   doctor (%)                                                     <0.001
#>      No                    1134 (  4.4)        57425 ( 15.5)
#>      Yes                  24384 ( 95.5)       313282 ( 84.4)
#>      NA                       6 (  0.0)          414 (  0.1)
#>   stress (%)                                                     <0.001
#>      Not too stressed     20041 ( 78.5)       266358 ( 71.8)
#>      stressed              5184 ( 20.3)        76986 ( 20.7)
#>      NA                     299 (  1.2)        27777 (  7.5)
#>   smoke (%)                                                      <0.001
#>      Current smoker        4481 ( 17.6)        93253 ( 25.1)
#>      Former smoker        13927 ( 54.6)       143421 ( 38.6)
#>      Never smoker          6981 ( 27.4)       132891 ( 35.8)
#>      NA                     135 (  0.5)         1556 (  0.4)
#>   drink (%)                                                      <0.001
#>      Current drinker      15852 ( 62.1)       279583 ( 75.3)
#>      Former driker         6820 ( 26.7)        48373 ( 13.0)
#>      Never drank           2421 (  9.5)        38195 ( 10.3)
#>      NA                     431 (  1.7)         4970 (  1.3)
#>   fruit (%)                                                      <0.001
#>      0-3 daily serving     4284 ( 16.8)        79088 ( 21.3)
#>      4-6 daily serving    10527 ( 41.2)       148684 ( 40.1)
#>      6+ daily serving      5047 ( 19.8)        73729 ( 19.9)
#>      NA                    5666 ( 22.2)        69620 ( 18.8)
#>   bp (%)                                                         <0.001
#>      No                   12611 ( 49.4)       315344 ( 85.0)
#>      Yes                  12857 ( 50.4)        55037 ( 14.8)
#>      NA                      56 (  0.2)          740 (  0.2)
#>   copd (%)                                                       <0.001
#>      No                   23378 ( 91.6)       267481 ( 72.1)
#>      Yes                   1449 (  5.7)         3043 (  0.8)
#>      NA                     697 (  2.7)       100597 ( 27.1)
#>   diab (%)                                                       <0.001
#>      No                   20461 ( 80.2)       353817 ( 95.3)
#>      Yes                   5038 ( 19.7)        17138 (  4.6)
```

```
#>      NA                        25 (  0.1)           166 (  0.0)
#>   province = South (%)     25271 ( 99.0)        363659 ( 98.0)     <0.001
#>   weight (mean (SD))      152.58 (181.69)       203.40 (244.28)    <0.001
#>   cycle (%)                                                        <0.001
#>      11                     7968 ( 31.2)        122798 ( 33.1)
#>      21                     9027 ( 35.4)        124838 ( 33.6)
#>      31                     8529 ( 33.4)        123485 ( 33.3)
#>   ID (mean (SD))       199839.07 (114705.35) 198466.74 (114661.51)  0.064
#>   OA (%)                                                           <0.001
#>      Control               12655 ( 49.6)        301675 ( 81.3)
#>      OA                     6522 ( 25.6)         34346 (  9.3)
#>      NA                     6347 ( 24.9)         35100 (  9.5)
#>   immigrate (%)                                                    <0.001
#>      > 10 years             2295 (  9.0)         24409 (  6.6)
#>      not immigrant         21342 ( 83.6)        316353 ( 85.2)
#>      recent                  159 (  0.6)         10476 (  2.8)
#>      NA                     1728 (  6.8)         19883 (  5.4)
#>   province.check (%)                                                  NaN
#>      NEWFOUNDLAND            519 (  2.0)          7398 (  2.0)
#>      PEI                     567 (  2.2)          7172 (  1.9)
#>      NOVA SCOTIA            1308 (  5.1)         14015 (  3.8)
#>      NEW BRUNSWICK          1223 (  4.8)         13786 (  3.7)
#>      QU\xc9BEC              1380 (  5.4)         20625 (  5.6)
#>      ONTARIO                8596 ( 33.7)        115053 ( 31.0)
#>      MANITOBA               1339 (  5.2)         22074 (  5.9)
#>      SASKATCHEWAN           1542 (  6.0)         21782 (  5.9)
#>      ALBERTA                1837 (  7.2)         38238 ( 10.3)
#>      BRITISH COLUMBIA       2847 ( 11.2)         46834 ( 12.6)
#>      YUKON/NWT/NUNAVT        173 (  0.7)          4884 (  1.3)
#>      NOT APPLICABLE            0 (  0.0)             0 (  0.0)
#>      DON'T KNOW                0 (  0.0)             0 (  0.0)
#>      REFUSAL                   0 (  0.0)             0 (  0.0)
#>      NOT STATED                0 (  0.0)             0 (  0.0)
#>      QUEBEC                 3839 ( 15.0)         52850 ( 14.2)
#>      NFLD & LAB.             274 (  1.1)          3832 (  1.0)
#>      YUKON/NWT/NUNA.          80 (  0.3)          2578 (  0.7)
CreateTableOne(data = analytic, strata = "OA", includeNA = TRUE)
#>                        Stratified by OA
#>                         Control               OA                    p      test
#>   n                        314542                40943
```

```
#>   CVD (%)                                                          <0.001
#>      event                 12655 (  4.0)          6522 ( 15.9)
#>      no event             301675 ( 95.9)         34346 ( 83.9)
#>      NA                      212 (  0.1)            75 (  0.2)
#>   age (%)                                                          <0.001
#>      20-29 years           46805 ( 14.9)           537 (  1.3)
#>      30-39 years           59233 ( 18.8)          1622 (  4.0)
#>      40-49 years           55598 ( 17.7)          4128 ( 10.1)
#>      50-59 years           43746 ( 13.9)          8994 ( 22.0)
#>      60-64 years           15772 (  5.0)          5100 ( 12.5)
#>      65 years and over     41661 ( 13.2)         20436 ( 49.9)
#>      teen                  51727 ( 16.4)           126 (  0.3)
#>   sex = Male (%)          153889 ( 48.9)         11627 ( 28.4)     <0.001
#>   married (%)                                                      <0.001
#>      not single           158065 ( 50.3)         21794 ( 53.2)
#>      single               155952 ( 49.6)         19099 ( 46.6)
#>      NA                      525 (  0.2)            50 (  0.1)
#>   race (%)                                                         <0.001
#>      Non-white             34028 ( 10.8)          1803 (  4.4)
#>      White                273378 ( 86.9)         38241 ( 93.4)
#>      NA                     7136 (  2.3)           899 (  2.2)
#>   edu (%)                                                          <0.001
#>      < 2ndary              92831 ( 29.5)         14539 ( 35.5)
#>      2nd grad.             52077 ( 16.6)          6291 ( 15.4)
#>      Other 2nd grad.       24099 (  7.7)          2484 (  6.1)
#>      Post-2nd grad.       140400 ( 44.6)         16887 ( 41.2)
#>      NA                     5135 (  1.6)           742 (  1.8)
#>   income (%)                                                       <0.001
#>      $29,999 or less       68530 ( 21.8)         16233 ( 39.6)
#>      $30,000-$49,999       61697 ( 19.6)          8360 ( 20.4)
#>      $50,000-$79,999       72657 ( 23.1)          6348 ( 15.5)
#>      $80,000 or more       67458 ( 21.4)          4191 ( 10.2)
#>      NA                    44200 ( 14.1)          5811 ( 14.2)
#>   bmi (%)                                                          <0.001
#>      Underweight            8660 (  2.8)           715 (  1.7)
#>      healthy weight       123416 ( 39.2)         12631 ( 30.9)
#>      Overweight           123898 ( 39.4)         20715 ( 50.6)
#>      NA                    58568 ( 18.6)          6882 ( 16.8)
#>   phyact (%)                                                       <0.001
#>      Active                84269 ( 26.8)          6968 ( 17.0)
```

```
#>      Inactive              143058 ( 45.5)          23604 ( 57.7)
#>      Moderate               75703 ( 24.1)           9176 ( 22.4)
#>      NA                     11512 (  3.7)           1195 (  2.9)
#>   doctor (%)                                                      <0.001
#>      No                     53335 ( 17.0)           2221 (  5.4)
#>      Yes                   260802 ( 82.9)          38717 ( 94.6)
#>      NA                       405 (  0.1)              5 (  0.0)
#>   stress (%)                                                      <0.001
#>      Not too stressed      223212 ( 71.0)          31769 ( 77.6)
#>      stressed               63923 ( 20.3)           8998 ( 22.0)
#>      NA                     27407 (  8.7)            176 (  0.4)
#>   smoke (%)                                                       <0.001
#>      Current smoker         79521 ( 25.3)           8087 ( 19.8)
#>      Former smoker         117745 ( 37.4)          20267 ( 49.5)
#>      Never smoker          116006 ( 36.9)          12428 ( 30.4)
#>      NA                      1270 (  0.4)            161 (  0.4)
#>   drink (%)                                                       <0.001
#>      Current drinker       239223 ( 76.1)          28622 ( 69.9)
#>      Former driker          37042 ( 11.8)           8668 ( 21.2)
#>      Never drank            34185 ( 10.9)           3128 (  7.6)
#>      NA                      4092 (  1.3)            525 (  1.3)
#>   fruit (%)                                                       <0.001
#>      0-3 daily serving      68629 ( 21.8)           6571 ( 16.0)
#>      4-6 daily serving     125177 ( 39.8)          17214 ( 42.0)
#>      6+ daily serving       62121 ( 19.7)           9123 ( 22.3)
#>      NA                     58615 ( 18.6)           8035 ( 19.6)
#>   bp (%)                                                          <0.001
#>      No                    275443 ( 87.6)          25551 ( 62.4)
#>      Yes                    38442 ( 12.2)          15341 ( 37.5)
#>      NA                       657 (  0.2)             51 (  0.1)
#>   copd (%)                                                        <0.001
#>      No                    213719 ( 67.9)          39007 ( 95.3)
#>      Yes                     2131 (  0.7)           1214 (  3.0)
#>      NA                     98692 ( 31.4)            722 (  1.8)
#>   diab (%)                                                        <0.001
#>      No                    301943 ( 96.0)          36211 ( 88.4)
#>      Yes                    12442 (  4.0)           4705 ( 11.5)
#>      NA                       157 (  0.0)             27 (  0.1)
#>   province = South (%)     307761 ( 97.8)          40507 ( 98.9)   <0.001
#>   weight (mean (SD))       211.50 (251.46)        159.00 (188.84)  <0.001
```

```
#>   cycle (%)                                                                <0.001
#>      11                    106231 ( 33.8)          12052 ( 29.4)
#>      21                    104530 ( 33.2)          14750 ( 36.0)
#>      31                    103781 ( 33.0)          14141 ( 34.5)
#>   ID (mean (SD))        197003.20 (115147.95) 204459.43 (113014.25) <0.001
#>   OA (%)                                                                    NaN
#>      Control               314542 (100.0)              0 (  0.0)
#>      OA                         0 (  0.0)          40943 (100.0)
#>      NA                         0 (  0.0)              0 (  0.0)
#>   immigrate (%)                                                            <0.001
#>      > 10 years             19385 (  6.2)           3622 (  8.8)
#>      not immigrant         268962 ( 85.5)          34509 ( 84.3)
#>      recent                 10187 (  3.2)            151 (  0.4)
#>      NA                     16008 (  5.1)           2661 (  6.5)
#>   province.check (%)                                                        NaN
#>      NEWFOUNDLAND            6315 (  2.0)            725 (  1.8)
#>      PEI                     5892 (  1.9)            817 (  2.0)
#>      NOVA SCOTIA            11081 (  3.5)           1880 (  4.6)
#>      NEW BRUNSWICK          11517 (  3.7)           1693 (  4.1)
#>      QU\xc9BEC              19111 (  6.1)           2035 (  5.0)
#>      ONTARIO                95651 ( 30.4)          13669 ( 33.4)
#>      MANITOBA               18050 (  5.7)           2272 (  5.5)
#>      SASKATCHEWAN           17941 (  5.7)           2166 (  5.3)
#>      ALBERTA                32207 ( 10.2)           3608 (  8.8)
#>      BRITISH COLUMBIA       40034 ( 12.7)           4873 ( 11.9)
#>      YUKON/NWT/NUNAVT        4446 (  1.4)            273 (  0.7)
#>      NOT APPLICABLE             0 (  0.0)              0 (  0.0)
#>      DON'T KNOW                 0 (  0.0)              0 (  0.0)
#>      REFUSAL                    0 (  0.0)              0 (  0.0)
#>      NOT STATED                 0 (  0.0)              0 (  0.0)
#>      QUEBEC                 46817 ( 14.9)           6366 ( 15.5)
#>      NFLD & LAB.             3145 (  1.0)            403 (  1.0)
#>      YUKON/NWT/NUNA.         2335 (  0.7)            163 (  0.4)

require(DataExplorer)
plot_missing(analytic)
```

Look for zero-cells

Creates two new variables based on age groups and generates summary tables. It also comments on the presence of 'zero cells' in one of the variables, which might require further handling.

```
analytic$age.65p <- analytic$age.teen <- 0
analytic$age.teen[analytic$age == "teen"] <- 1
analytic$age.65p[analytic$age == "65 years and over"] <- 1
CreateTableOne(data = analytic, strata = "age.teen", includeNA = TRUE)
#>                          Stratified by age.teen
#>                           0                     1                     p      test
#>   n                          344786                 52387
#>   CVD (%)                                                            <0.001
#>      event                    25259 ( 7.3)            265 (  0.5)
#>      no event                319031 (92.5)          52090 ( 99.4)
#>      NA                         496 ( 0.1)             32 (  0.1)
#>   age (%)                                                            <0.001
#>      20-29 years              48652 (14.1)              0 (  0.0)
#>      30-39 years              63810 (18.5)              0 (  0.0)
#>      40-49 years              65111 (18.9)              0 (  0.0)
#>      50-59 years              61035 (17.7)              0 (  0.0)
#>      60-64 years              25265 ( 7.3)              0 (  0.0)
#>      65 years and over        80913 (23.5)              0 (  0.0)
#>      teen                         0 ( 0.0)          52387 (100.0)
```

```
#>   sex = Male (%)          155980 (45.2)          26543 ( 50.7)     <0.001
#>   married (%)                                                      <0.001
#>      not single           201528 (58.5)            685 (  1.3)
#>      single               142639 (41.4)          51660 ( 98.6)
#>      NA                      619 ( 0.2)             42 (  0.1)
#>   race (%)                                                         <0.001
#>      Non-white             31107 ( 9.0)           7534 ( 14.4)
#>      White                305497 (88.6)          43725 ( 83.5)
#>      NA                     8182 ( 2.4)           1128 (  2.2)
#>   edu (%)                                                          <0.001
#>      < 2ndary              83649 (24.3)          40776 ( 77.8)
#>      2nd grad.             59205 (17.2)           5548 ( 10.6)
#>      Other 2nd grad.       24580 ( 7.1)           4420 (  8.4)
#>      Post-2nd grad.       170707 (49.5)           1265 (  2.4)
#>      NA                     6645 ( 1.9)            378 (  0.7)
#>   income (%)                                                       <0.001
#>      $29,999 or less       93630 (27.2)           7701 ( 14.7)
#>      $30,000-$49,999       69798 (20.2)           8142 ( 15.5)
#>      $50,000-$79,999       73596 (21.3)          11512 ( 22.0)
#>      $80,000 or more       63697 (18.5)          12018 ( 22.9)
#>      NA                    44065 (12.8)          13014 ( 24.8)
#>   bmi (%)                                                          <0.001
#>      Underweight            7277 ( 2.1)           2839 (  5.4)
#>      healthy weight       138611 (40.2)           9922 ( 18.9)
#>      Overweight           163701 (47.5)           2520 (  4.8)
#>      NA                    35197 (10.2)          37106 ( 70.8)
#>   phyact (%)                                                       <0.001
#>      Active                74738 (21.7)          23833 ( 45.5)
#>      Inactive             176573 (51.2)          14166 ( 27.0)
#>      Moderate              82158 (23.8)          11349 ( 21.7)
#>      NA                    11317 ( 3.3)           3039 (  5.8)
#>   doctor (%)                                                       <0.001
#>      No                    49874 (14.5)           8749 ( 16.7)
#>      Yes                  294763 (85.5)          43342 ( 82.7)
#>      NA                       149 ( 0.0)            296 (  0.6)
#>   stress (%)                                                       <0.001
#>      Not too stressed     265391 (77.0)          21353 ( 40.8)
#>      stressed              78044 (22.6)           4253 (  8.1)
#>      NA                     1351 ( 0.4)          26781 ( 51.1)
#>   smoke (%)                                                        <0.001
```

```
#>      Current smoker        88986 (25.8)           8866 ( 16.9)
#>      Former smoker        150004 (43.5)           7566 ( 14.4)
#>      Never smoker         104332 (30.3)          35685 ( 68.1)
#>      NA                     1464 ( 0.4)            270 (  0.5)
#>   drink (%)                                                          <0.001
#>      Current drinker      268269 (77.8)          27464 ( 52.4)
#>      Former driker         50929 (14.8)           4370 (  8.3)
#>      Never drank           20754 ( 6.0)          19916 ( 38.0)
#>      NA                     4834 ( 1.4)            637 (  1.2)
#>   fruit (%)                                                          <0.001
#>      0-3 daily serving     72392 (21.0)          11049 ( 21.1)
#>      4-6 daily serving    139753 (40.5)          19627 ( 37.5)
#>      6+ daily serving      66831 (19.4)          12026 ( 23.0)
#>      NA                    65810 (19.1)           9685 ( 18.5)
#>   bp (%)                                                             <0.001
#>      No                   276318 (80.1)          51846 ( 99.0)
#>      Yes                   67763 (19.7)            308 (  0.6)
#>      NA                      705 ( 0.2)            233 (  0.4)
#>   copd (%)                                                           <0.001
#>      No                   291191 (84.5)              0 (  0.0)
#>      Yes                    4508 ( 1.3)              0 (  0.0)
#>      NA                    49087 (14.2)          52387 (100.0)
#>   diab (%)                                                           <0.001
#>      No                   322448 (93.5)          52141 ( 99.5)
#>      Yes                   22032 ( 6.4)            199 (  0.4)
#>      NA                      306 ( 0.1)             47 (  0.1)
#>   province = South (%)    338450 (98.2)          51001 ( 97.4)      <0.001
#>   weight (mean (SD))      201.76 (245.97)       189.09 (205.24)     <0.001
#>   cycle (%)                                                          <0.001
#>      11                   113323 (32.9)          17557 ( 33.5)
#>      21                   115548 (33.5)          18524 ( 35.4)
#>      31                   115915 (33.6)          16306 ( 31.1)
#>   ID (mean (SD))       199143.77 (114810.36) 194922.59 (113553.38) <0.001
#>   OA (%)                                                             <0.001
#>      Control              262815 (76.2)          51727 ( 98.7)
#>      OA                    40817 (11.8)            126 (  0.2)
#>      NA                    41154 (11.9)            534 (  1.0)
#>   immigrate (%)                                                      <0.001
#>      > 10 years            25976 ( 7.5)            770 (  1.5)
#>      not immigrant        289651 (84.0)          48427 ( 92.4)
```

```
#>      recent                  8710 ( 2.5)           1934 (  3.7)
#>      NA                     20449 ( 5.9)           1256 (  2.4)
#>   province.check (%)                                                    NaN
#>      NEWFOUNDLAND            6646 ( 1.9)           1278 (  2.4)
#>      PEI                     6802 ( 2.0)            942 (  1.8)
#>      NOVA SCOTIA            13337 ( 3.9)           2004 (  3.8)
#>      NEW BRUNSWICK          13057 ( 3.8)           1968 (  3.8)
#>      QU\xc9BEC              19186 ( 5.6)           2826 (  5.4)
#>      ONTARIO               107768 (31.3)          16053 ( 30.6)
#>      MANITOBA               20362 ( 5.9)           3092 (  5.9)
#>      SASKATCHEWAN           20160 ( 5.8)           3201 (  6.1)
#>      ALBERTA                34293 ( 9.9)           5834 ( 11.1)
#>      BRITISH COLUMBIA       43431 (12.6)           6336 ( 12.1)
#>      YUKON/NWT/NUNAVT        4170 ( 1.2)            894 (  1.7)
#>      NOT APPLICABLE             0 ( 0.0)              0 (  0.0)
#>      DON'T KNOW                 0 ( 0.0)              0 (  0.0)
#>      REFUSAL                    0 ( 0.0)              0 (  0.0)
#>      NOT STATED                 0 ( 0.0)              0 (  0.0)
#>      QUEBEC                 49806 (14.4)           6958 ( 13.3)
#>      NFLD & LAB.             3602 ( 1.0)            509 (  1.0)
#>      YUKON/NWT/NUNA.         2166 ( 0.6)            492 (  0.9)
#>   age.teen (mean (SD))       0.00 (0.00)           1.00 (0.00)      <0.001
#>   age.65p (mean (SD))        0.23 (0.42)           0.00 (0.00)      <0.001
# copd has zero cells
# analytic$age[analytic$age == 'teen'] <- NA (will set this if we use copd)
```

```r
CreateTableOne(data = analytic, strata = "age.65p", includeNA = TRUE)
#>                        Stratified by age.65p
#>                         0                     1                     p      test
#>   n                        316260                 80913
#>   CVD (%)                                                            <0.001
#>      event                    9028 ( 2.9)          16496 ( 20.4)
#>      no event               306923 (97.0)          64198 ( 79.3)
#>      NA                        309 ( 0.1)            219 (  0.3)
#>   age (%)                                                            <0.001
#>      20-29 years            48652 (15.4)              0 (  0.0)
#>      30-39 years            63810 (20.2)              0 (  0.0)
#>      40-49 years            65111 (20.6)              0 (  0.0)
#>      50-59 years            61035 (19.3)              0 (  0.0)
#>      60-64 years            25265 ( 8.0)              0 (  0.0)
```

```
#>      65 years and over          0 ( 0.0)          80913 (100.0)
#>      teen                   52387 (16.6)              0 (  0.0)
#>   sex = Male (%)           150152 (47.5)          32371 ( 40.0)     <0.001
#>   married (%)                                                       <0.001
#>      not single            163660 (51.7)          38553 ( 47.6)
#>      single                152077 (48.1)          42222 ( 52.2)
#>      NA                       523 ( 0.2)            138 (  0.2)
#>   race (%)                                                          <0.001
#>      Non-white              35329 (11.2)           3312 (  4.1)
#>      White                 274000 (86.6)          75222 ( 93.0)
#>      NA                      6931 ( 2.2)           2379 (  2.9)
#>   edu (%)                                                           <0.001
#>      < 2ndary               84832 (26.8)          39593 ( 48.9)
#>      2nd grad.              53974 (17.1)          10779 ( 13.3)
#>      Other 2nd grad.        25305 ( 8.0)           3695 (  4.6)
#>      Post-2nd grad.        147385 (46.6)          24587 ( 30.4)
#>      NA                      4764 ( 1.5)           2259 (  2.8)
#>   income (%)                                                        <0.001
#>      $29,999 or less        62513 (19.8)          38818 ( 48.0)
#>      $30,000-$49,999        62296 (19.7)          15644 ( 19.3)
#>      $50,000-$79,999        77283 (24.4)           7825 (  9.7)
#>      $80,000 or more        72566 (22.9)           3149 (  3.9)
#>      NA                     41602 (13.2)          15477 ( 19.1)
#>   bmi (%)                                                           <0.001
#>      Underweight             8588 ( 2.7)           1528 (  1.9)
#>      healthy weight        124932 (39.5)          23601 ( 29.2)
#>      Overweight            136225 (43.1)          29996 ( 37.1)
#>      NA                     46515 (14.7)          25788 ( 31.9)
#>   phyact (%)                                                        <0.001
#>      Active                 85384 (27.0)          13187 ( 16.3)
#>      Inactive              144019 (45.5)          46720 ( 57.7)
#>      Moderate               76602 (24.2)          16905 ( 20.9)
#>      NA                     10255 ( 3.2)           4101 (  5.1)
#>   doctor (%)                                                        <0.001
#>      No                     53972 (17.1)           4651 (  5.7)
#>      Yes                   261866 (82.8)          76239 ( 94.2)
#>      NA                       422 ( 0.1)             23 (  0.0)
#>   stress (%)                                                        <0.001
#>      Not too stressed      215454 (68.1)          71290 ( 88.1)
#>      stressed               73402 (23.2)           8895 ( 11.0)
```

```
#>      NA                    27404 ( 8.7)            728 (  0.9)
#>   smoke (%)                                                        <0.001
#>      Current smoker        88068 (27.8)           9784 ( 12.1)
#>      Former smoker        115111 (36.4)          42459 ( 52.5)
#>      Never smoker         111879 (35.4)          28138 ( 34.8)
#>      NA                     1202 ( 0.4)            532 (  0.7)
#>   drink (%)                                                        <0.001
#>      Current drinker      245156 (77.5)          50577 ( 62.5)
#>      Former driker         35401 (11.2)          19898 ( 24.6)
#>      Never drank           31888 (10.1)           8782 ( 10.9)
#>      NA                     3815 ( 1.2)           1656 (  2.0)
#>   fruit (%)                                                        <0.001
#>      0-3 daily serving     72908 (23.1)          10533 ( 13.0)
#>      4-6 daily serving    124621 (39.4)          34759 ( 43.0)
#>      6+ daily serving      61855 (19.6)          17002 ( 21.0)
#>      NA                    56876 (18.0)          18619 ( 23.0)
#>   bp (%)                                                           <0.001
#>      No                   282174 (89.2)          45990 ( 56.8)
#>      Yes                   33346 (10.5)          34725 ( 42.9)
#>      NA                      740 ( 0.2)            198 (  0.2)
#>   copd (%)                                                         <0.001
#>      No                   213221 (67.4)          77970 ( 96.4)
#>      Yes                    1791 ( 0.6)           2717 (  3.4)
#>      NA                   101248 (32.0)            226 (  0.3)
#>   diab (%)                                                         <0.001
#>      No                   305027 (96.4)          69562 ( 86.0)
#>      Yes                   10974 ( 3.5)          11257 ( 13.9)
#>      NA                      259 ( 0.1)             94 (  0.1)
#>   province = South (%)    309016 (97.7)          80435 ( 99.4)     <0.001
#>   weight (mean (SD))      215.36 (255.33)       140.38 (160.88)    <0.001
#>   cycle (%)                                                        <0.001
#>      11                   106647 (33.7)          24233 ( 29.9)
#>      21                   105506 (33.4)          28566 ( 35.3)
#>      31                   104107 (32.9)          28114 ( 34.7)
#>   ID (mean (SD))       197072.98 (115035.66) 204504.77 (112956.66) <0.001
#>   OA (%)                                                           <0.001
#>      Control              272881 (86.3)          41661 ( 51.5)
#>      OA                    20507 ( 6.5)          20436 ( 25.3)
#>      NA                    22872 ( 7.2)          18816 ( 23.3)
#>   immigrate (%)                                                    <0.001
```

```
#>      > 10 years            17607 ( 5.6)           9139 ( 11.3)
#>      not immigrant        273622 (86.5)          64456 ( 79.7)
#>      recent                10325 ( 3.3)            319 (  0.4)
#>      NA                    14706 ( 4.6)           6999 (  8.7)
#>   province.check (%)                                                    NaN
#>      NEWFOUNDLAND           6665 ( 2.1)           1259 (  1.6)
#>      PEI                    5993 ( 1.9)           1751 (  2.2)
#>      NOVA SCOTIA           11896 ( 3.8)           3445 (  4.3)
#>      NEW BRUNSWICK         11856 ( 3.7)           3169 (  3.9)
#>      QU\xc9BEC             18128 ( 5.7)           3884 (  4.8)
#>      ONTARIO               97660 (30.9)          26161 ( 32.3)
#>      MANITOBA              17967 ( 5.7)           5487 (  6.8)
#>      SASKATCHEWAN          17507 ( 5.5)           5854 (  7.2)
#>      ALBERTA               33445 (10.6)           6682 (  8.3)
#>      BRITISH COLUMBIA      39394 (12.5)          10373 ( 12.8)
#>      YUKON/NWT/NUNAVT       4765 ( 1.5)            299 (  0.4)
#>      NOT APPLICABLE            0 ( 0.0)              0 (  0.0)
#>      DON'T KNOW                0 ( 0.0)              0 (  0.0)
#>      REFUSAL                   0 ( 0.0)              0 (  0.0)
#>      NOT STATED                0 ( 0.0)              0 (  0.0)
#>      QUEBEC                45226 (14.3)          11538 ( 14.3)
#>      NFLD & LAB.            3279 ( 1.0)            832 (  1.0)
#>      YUKON/NWT/NUNA.        2479 ( 0.8)            179 (  0.2)
#>   age.teen (mean (SD))      0.17 (0.37)           0.00 (0.00)      <0.001
#>   age.65p (mean (SD))       0.00 (0.00)           1.00 (0.00)      <0.001
analytic$age.65p <- analytic$age.teen <- NULL
```

Produces frequency tables for multiple variable combinations to check the distribution of the data and identify issues.

```
table(analytic$province.check,analytic$fruit)
#>
#>                     0-3 daily serving 4-6 daily serving 6+ daily serving
#>   NEWFOUNDLAND                   2991              3401             1237
#>   PEI                            2280              3654             1506
#>   NOVA SCOTIA                    3089              4804             1974
#>   NEW BRUNSWICK                  2989              4730             1880
#>   QU\xc9BEC                      5568             10502             5786
#>   ONTARIO                       28752             59466            30746
#>   MANITOBA                       4561              7669             3095
```

```
#>   SASKATCHEWAN                  4173              7390              3003
#>   ALBERTA                      10828             18901              8251
#>   BRITISH COLUMBIA             10726             24422             12390
#>   YUKON/NWT/NUNAVT              1829              2045              1023
#>   NOT APPLICABLE                   0                 0                 0
#>   DON'T KNOW                       0                 0                 0
#>   REFUSAL                          0                 0                 0
#>   NOT STATED                       0                 0                 0
#>   QUEBEC                        5655             12396              7966
#>   NFLD & LAB.                      0                 0                 0
#>   YUKON/NWT/NUNA.                  0                 0                 0
table(analytic$age)
#>
#>       20-29 years       30-39 years       40-49 years       50-59 years
#>             48652             63810             65111             61035
#>       60-64 years 65 years and over              teen
#>             25265             80913             52387
table(analytic$copd, analytic$age)
#>
#>       20-29 years 30-39 years 40-49 years 50-59 years 60-64 years
#>   No            0       63645       64735       60203       24638
#>   Yes           0         117         320         768         586
#>
#>       65 years and over  teen
#>   No              77970     0
#>   Yes              2717     0
table(analytic$stress, analytic$age)
#>
#>                    20-29 years 30-39 years 40-49 years 50-59 years 60-64 years
#>   Not too stressed       37117       45494       45316       45226       20948
#>   stressed               11472       18197       19639       15623        4218
#>
#>                    65 years and over  teen
#>   Not too stressed             71290 21353
#>   stressed                      8895  4253
```

- universe 15 + is not an issue for `stress` as age starts from 20
- `copd` is problematic!

Creates tables to look at the distribution of a specific variable

across different cycles (time periods) of the survey. Notes differences and issues.

- `fruit` variable measured in an **optional component** (not available in all cycles)

```r
table(analytic$province.check[analytic$cycle==11],
      analytic$fruit[analytic$cycle==11])
#>
#>                    0-3 daily serving 4-6 daily serving 6+ daily serving
#>   NEWFOUNDLAND                  1512              1643              689
#>   PEI                           1084              1738              773
#>   NOVA SCOTIA                   1732              2500             1036
#>   NEW BRUNSWICK                 1663              2363              934
#>   QU\xc9BEC                     5568             10502             5786
#>   ONTARIO                      10437             19478             8809
#>   MANITOBA                      2604              4214             1526
#>   SASKATCHEWAN                  2386              3957             1387
#>   ALBERTA                       4391              7050             2664
#>   BRITISH COLUMBIA              4321              9350             4278
#>   YUKON/NWT/NUNAVT               999              1014              448
#>   NOT APPLICABLE                   0                 0                0
#>   DON'T KNOW                       0                 0                0
#>   REFUSAL                          0                 0                0
#>   NOT STATED                       0                 0                0
#>   QUEBEC                           0                 0                0
#>   NFLD & LAB.                      0                 0                0
#>   YUKON/NWT/NUNA.                  0                 0                0
```

```r
table(analytic$province.check[analytic$cycle==21],
      analytic$fruit[analytic$cycle==21])
#>
#>                    0-3 daily serving 4-6 daily serving 6+ daily serving
#>   NEWFOUNDLAND                  1479              1758              548
#>   PEI                            615               947              344
#>   NOVA SCOTIA                   1357              2304              938
#>   NEW BRUNSWICK                 1326              2367              946
#>   QU\xc9BEC                        0                 0                0
#>   ONTARIO                       9365             20356            10933
#>   MANITOBA                      1957              3455             1569
#>   SASKATCHEWAN                  1787              3433             1616
```

```
#>   ALBERTA                        3326              6376              3046
#>   BRITISH COLUMBIA               3186              7727              4224
#>   YUKON/NWT/NUNAVT                830              1031               575
#>   NOT APPLICABLE                    0                 0                 0
#>   DON'T KNOW                        0                 0                 0
#>   REFUSAL                           0                 0                 0
#>   NOT STATED                        0                 0                 0
#>   QUEBEC                         5655             12396              7966
#>   NFLD & LAB.                       0                 0                 0
#>   YUKON/NWT/NUNA.                   0                 0                 0
# a different QUEBEC spelling used
```

```
table(analytic$province.check[analytic$cycle==31],
      analytic$fruit[analytic$cycle==31])
#>
#>                     0-3 daily serving 4-6 daily serving 6+ daily serving
#>   NEWFOUNDLAND                      0                 0                 0
#>   PEI                             581               969               389
#>   NOVA SCOTIA                       0                 0                 0
#>   NEW BRUNSWICK                     0                 0                 0
#>   QU\xc9BEC                         0                 0                 0
#>   ONTARIO                        8950             19632             11004
#>   MANITOBA                          0                 0                 0
#>   SASKATCHEWAN                      0                 0                 0
#>   ALBERTA                        3111              5475              2541
#>   BRITISH COLUMBIA               3219              7345              3888
#>   YUKON/NWT/NUNAVT                  0                 0                 0
#>   NOT APPLICABLE                    0                 0                 0
#>   DON'T KNOW                        0                 0                 0
#>   REFUSAL                           0                 0                 0
#>   NOT STATED                        0                 0                 0
#>   QUEBEC                            0                 0                 0
#>   NFLD & LAB.                       0                 0                 0
#>   YUKON/NWT/NUNA.                   0                 0                 0
# The real problem!
```

- Look at data dictionaries in all cycles
  - cycle 1.1 FVCADTOT Universe: All respondents
  - cycle 2.1 FVCCDTOT Universe: All respondents

- cycle 3.1 FVCEDTOT Universe: Respondents with FVCEFOPT = 1

Below we delete or modify problematic data, and removes unnecessary variables. Checks the dimensions before and after data cleanup.

```
dim(analytic)
#> [1] 397173     24
analytic1 <- analytic
# analytic1$South[analytic1$province.check == "NFLD & LAB."] <- NA
# analytic1$South[analytic1$province.check == "YUKON/NWT/NUNA."] <- NA
# analytic1 <- subset(analytic, province.check != "NFLD & LAB." &
#                       province.check != "YUKON/NWT/NUNA." )
dim(analytic1)
#> [1] 397173     24

analytic1$copd <- NULL # will bring this later for missing data analysis
# CreateTableOne(data = analytic1, strata = "OA", includeNA = TRUE)
# analytic1 <- droplevels.data.frame(analytic1)
analytic1$province.check <- NULL # we already have simplified province variable
# CreateTableOne(data = analytic1, strata = "OA", includeNA = TRUE)
```

Set appropriate reference

Save the original data (with missing values)!

```
analytic.miss <- analytic1
```

Relevels factors in the dataset so that a specific level is set as the reference level. This is often needed for statistical analysis.

```
analytic.miss$smoke <- relevel(as.factor(analytic.miss$smoke), ref='Never smoker')
analytic.miss$drink <- relevel(as.factor(analytic.miss$drink), ref='Never drank')
analytic.miss$province <- relevel(as.factor(analytic.miss$province), ref='South')
analytic.miss$immigrate <- relevel(as.factor(analytic.miss$immigrate), ref='not immigrant')
```

Complete data options

Creates a new dataset that omits all rows containing any missing values. This is generally not recommended for most data analysis, as it can introduce bias.

```
# Wrong thing to do for survey data analysis!!
analytic2 <- as.data.frame(na.omit(analytic1))
dim(analytic2)
#> [1] 185613     22
# tab1 <- CreateTableOne(data = analytic2, strata = "OA", includeNA = TRUE)
# print(tab1, test=FALSE, showAllLevels = TRUE)
```

Saving dataset

Let us check the dimensions of multiple data objects and then save them to a file for future use.

```
dim(cc123a)
#> [1] 397173     25
dim(analytic)
#> [1] 397173     24
dim(analytic.miss)
#> [1] 397173     22
dim(analytic2)
#> [1] 185613     22
save(analytic.miss, analytic2, file = "Data/surveydata/cchs123b.RData")
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

CCHS: Bivariate analysis

The following tutorial is performing bivariate analysis on our CCHS analytic dataset to examine relationships between two variables (association question).

We load several R packages required for bivariate analysis, statistical tests, and data visualization.

```
# Load required packages
library(survey)
library(knitr)
library(car)
library(tableone)
library(DataExplorer)
library(Publish)
library(ROCR)
library(WeightedROC)
library(jtools)
```

Load data

We load the dataset into the R environment and lists all available variables and objects.

```
load("Data/surveydata/cchs123b.RData")
ls()
#> [1] "analytic.miss"   "analytic2"       "has_annotations"
```

Preparing data

Weights

Here, the weights of survey respondents are accumulated, to account for the combination of different cycles of the data.

```
analytic.miss$weight <- analytic.miss$weight/3 # 3 cycles combined
```

Fixing variable types

We convert several variables to categorical or "factor" types, which are better suited for some statistical analysis when variables have categories.

```
var.names <- c("CVD", "age", "sex", "married", "race", "edu", "income", "bmi",
               "phyact", "doctor", "stress", "smoke", "drink", "fruit", "bp",
               "diab", "province", "OA", "immigrate")
analytic.miss[var.names] <- lapply(analytic.miss[var.names] , factor)
str(analytic.miss)
#> 'data.frame':    397173 obs. of  22 variables:
#>  $ CVD      : Factor w/ 2 levels "event","no event": 1 2 2 2 2 2 2 2 2 2 ...
#>  $ age      : Factor w/ 7 levels "20-29 years",..: 6 6 2 6 1 6 3 7 1 1 ...
#>  $ sex      : Factor w/ 2 levels "Female","Male": 1 1 2 1 1 2 2 2 1 2 ...
#>  $ married  : Factor w/ 2 levels "not single","single": 2 2 1 2 2 1 1 2 2 2 ...
#>  $ race     : Factor w/ 2 levels "Non-white","White": 2 2 2 2 2 2 2 2 2 2 ...
#>  $ edu      : Factor w/ 4 levels "< 2ndary","2nd grad.",..: 2 4 4 4 4 4 4 1 4 4 ...
#>  $ income   : Factor w/ 4 levels "$29,999 or less",..: 1 1 4 1 2 2 1 1 NA 4 ...
#>  $ bmi      : Factor w/ 3 levels "Underweight",..: NA NA 2 NA 2 NA 3 NA 2 3 ...
#>  $ phyact   : Factor w/ 3 levels "Active","Inactive",..: 2 2 2 2 2 2 1 1 2 3 ...
#>  $ doctor   : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
#>  $ stress   : Factor w/ 2 levels "Not too stressed",..: 1 1 2 1 1 1 1 NA 1 1 ...
#>  $ smoke    : Factor w/ 3 levels "Never smoker",..: 3 1 3 3 2 2 3 1 2 2 ...
#>  $ drink    : Factor w/ 3 levels "Never drank",..: 2 1 2 2 2 2 3 1 2 2 ...
#>  $ fruit    : Factor w/ 3 levels "0-3 daily serving",..: 2 2 2 3 3 3 2 2 2 2 ...
#>  $ bp       : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
#>  $ diab     : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
#>  $ province : Factor w/ 2 levels "South","North": 1 1 1 1 1 1 1 1 1 1 ...
#>  $ weight   : num  47.6 23.8 56.1 23.8 65.4 ...
```

```
#>  $ cycle    : Factor w/ 3 levels "11","21","31": 1 1 1 1 1 1 1 1 1 1 ...
#>  $ ID       : int  1 2 3 4 5 6 7 8 9 10 ...
#>  $ OA       : Factor w/ 2 levels "Control","OA": 1 1 1 1 1 1 1 1 1 1 ...
#>  $ immigrate: Factor w/ 3 levels "not immigrant",..: 1 1 3 1 1 1 1 1 1 1 ...
```

The code identifies rows where data is missing and labels them for later analyses.

```
analytic.miss$miss <- 1
head(analytic.miss$ID) # full data
#> [1] 1 2 3 4 5 6
head(analytic2$ID) # complete case
#> [1]  3  5  7 10 11 13
head(analytic.miss$ID[analytic.miss$ID %in% analytic2$ID])
#> [1]  3  5  7 10 11 13
analytic.miss$miss[analytic.miss$ID %in% analytic2$ID] <- 0
table(analytic.miss$miss)
#>
#>      0      1
#> 185613 211560
```

Setting Design

The code sets up the survey design, specifying weights (but no specific clustering and stratification, as they are unavailable for CCHS public access data), for use in survey-weighted analyses.

```
require(survey)
summary(analytic.miss$weight)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    0.39   21.76   42.21   66.70   81.07 2384.98
w.design0 <- svydesign(id=~1, weights=~weight,
                       data=analytic.miss)
summary(weights(w.design0))
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    0.39   21.76   42.21   66.70   81.07 2384.98
sd(weights(w.design0))
#> [1] 80.34263
```

This creates a subset of the data where there are no missing values. Note that subset was done to the design object `w.design0`, not the data `analytic.miss`.

```
w.design <- subset(w.design0, miss == 0)
summary(weights(w.design))
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    0.39   23.85   45.98   71.54   87.30 2384.98
sd(weights(w.design))
#> [1] 84.97819
```

Bivariate analysis

Table 1 (weighted)

0.0.0.0.1 * Stratified by exposure

These tables contain descriptive statistics, stratified by different categories. They can be useful for understanding how variables relate to the exposure or outcome in the data.

```
require(tableone)
var.names <- c("CVD", "age", "sex", "married", "race", "edu", "income", "bmi",
               "phyact", "doctor", "stress", "smoke", "drink", "fruit", "bp",
               "diab", "province", "immigrate") # exclude "OA"
# tab1 <- CreateTableOne(var = var.names, strata= "OA", data=analytic.miss, test = TRUE)
# print(tab1)
tab2 <- svyCreateTableOne(var = var.names, strata= "OA",
                          data=w.design, test = TRUE)
print(tab2)
#>                        Stratified by OA
#>                         Control            OA                p      test
#>   n                     12124961.5         1153392.2
#>   CVD = no event (%)    11786450.6 (97.2)  1020965.5 (88.5)  <0.001
#>   age (%)                                                    <0.001
#>      20-29 years         2668880.8 (22.0)    28317.5 ( 2.5)
#>      30-39 years         3009426.7 (24.8)    77159.3 ( 6.7)
#>      40-49 years         3108300.9 (25.6)   211515.1 (18.3)
#>      50-59 years         1900845.3 (15.7)   350264.7 (30.4)
#>      60-64 years          546041.8 ( 4.5)   163724.7 (14.2)
```

```
#>      65 years and over    624706.7 ( 5.2)   322033.4 (27.9)
#>      teen                 266759.3 ( 2.2)      377.4 ( 0.0)
#>   sex = Male (%)         6374765.5 (52.6)   379850.5 (32.9)  <0.001
#>   married = single (%)   4120611.0 (34.0)   367647.7 (31.9)  <0.001
#>   race = White (%)      10312228.2 (85.0)  1081778.6 (93.8)  <0.001
#>   edu (%)                                                     <0.001
#>      < 2ndary            1752318.3 (14.5)   309652.8 (26.8)
#>      2nd grad.           2314713.1 (19.1)   203437.5 (17.6)
#>      Other 2nd grad.     1078645.2 ( 8.9)    79255.1 ( 6.9)
#>      Post-2nd grad.      6979284.9 (57.6)   561046.8 (48.6)
#>   income (%)                                                  <0.001
#>      $29,999 or less     2051640.6 (16.9)   353862.9 (30.7)
#>      $30,000-$49,999     2436063.7 (20.1)   272484.1 (23.6)
#>      $50,000-$79,999     3495902.5 (28.8)   275115.8 (23.9)
#>      $80,000 or more     4141354.6 (34.2)   251929.4 (21.8)
#>   bmi (%)                                                     <0.001
#>      Underweight          346004.9 ( 2.9)    22064.6 ( 1.9)
#>      healthy weight      6019004.1 (49.6)   431570.2 (37.4)
#>      Overweight          5759952.5 (47.5)   699757.4 (60.7)
#>   phyact (%)                                                  <0.001
#>      Active              3037314.2 (25.1)   216879.5 (18.8)
#>      Inactive            5982492.3 (49.3)   647856.2 (56.2)
#>      Moderate            3105154.9 (25.6)   288656.5 (25.0)
#>   doctor = Yes (%)      10087473.8 (83.2)  1090763.9 (94.6)  <0.001
#>   stress = stressed (%)  3123770.9 (25.8)   301895.2 (26.2)   0.420
#>   smoke (%)                                                   <0.001
#>      Never smoker        4043479.9 (33.3)   320323.7 (27.8)
#>      Current smoker      3219168.6 (26.5)   275835.7 (23.9)
#>      Former smoker       4862313.0 (40.1)   557232.7 (48.3)
#>   drink (%)                                                   <0.001
#>      Never drank          678435.7 ( 5.6)    66085.0 ( 5.7)
#>      Current drinker    10297713.4 (84.9)   887808.2 (77.0)
#>      Former driker       1148812.3 ( 9.5)   199498.9 (17.3)
#>   fruit (%)                                                   <0.001
#>      0-3 daily serving   3214156.0 (26.5)   236483.8 (20.5)
#>      4-6 daily serving   6001124.3 (49.5)   588323.8 (51.0)
#>      6+ daily serving    2909681.1 (24.0)   328584.5 (28.5)
#>   bp = Yes (%)           1212548.2 (10.0)   347269.8 (30.1)  <0.001
#>   diab = Yes (%)          377876.2 ( 3.1)   104541.0 ( 9.1)  <0.001
#>   province = North (%)     27124.3 ( 0.2)     1825.1 ( 0.2)  <0.001
```

```
#>   immigrate (%)                                                  <0.001
#>      not immigrant       9898636.6 (81.6)   994682.5 (86.2)
#>      > 10 years          1384672.6 (11.4)   146879.8 (12.7)
#>      recent               841652.3 ( 6.9)    11829.8 ( 1.0)
```

0.0.0.0.2 * Stratified by outcome

This table is generally useful for logistic regression analysis

```
var.names <- c("OA", "age", "sex", "married", "race", "edu", "income", "bmi",
               "phyact", "doctor", "stress", "smoke", "drink", "fruit", "bp",
               "diab", "province", "immigrate") # exclude "CVD"
tab3 <- svyCreateTableOne(var = var.names, strata= "CVD", data=w.design, test = TRUE)
print(tab3)
#>                         Stratified by CVD
#>                          event            no event           p      test
#>   n                     470937.5         12807416.1
#>   OA = OA (%)           132426.7 (28.1)   1020965.5 ( 8.0)  <0.001
#>   age (%)                                                    <0.001
#>      20-29 years         14966.0 ( 3.2)   2682232.3 (20.9)
#>      30-39 years         24105.5 ( 5.1)   3062480.5 (23.9)
#>      40-49 years         63520.1 (13.5)   3256296.0 (25.4)
#>      50-59 years        122613.0 (26.0)   2128497.0 (16.6)
#>      60-64 years         72328.3 (15.4)    637438.1 ( 5.0)
#>      65 years and over  172742.2 (36.7)    773997.8 ( 6.0)
#>      teen                   662.3 ( 0.1)    266474.4 ( 2.1)
#>   sex = Male (%)        267743.8 (56.9)   6486872.2 (50.6)  <0.001
#>   married = single (%)  143747.0 (30.5)   4344511.8 (33.9)  <0.001
#>   race = White (%)      434591.6 (92.3)  10959415.2 (85.6)  <0.001
#>   edu (%)                                                    <0.001
#>      < 2ndary           147338.4 (31.3)   1914632.6 (14.9)
#>      2nd grad.           77705.6 (16.5)   2440445.0 (19.1)
#>      Other 2nd grad.     30921.3 ( 6.6)   1126979.0 ( 8.8)
#>      Post-2nd grad.     214972.2 (45.6)   7325359.4 (57.2)
#>   income (%)                                                 <0.001
#>      $29,999 or less    164929.4 (35.0)   2240574.2 (17.5)
#>      $30,000-$49,999    109988.2 (23.4)   2598559.6 (20.3)
#>      $50,000-$79,999    103091.1 (21.9)   3667927.1 (28.6)
#>      $80,000 or more     92928.7 (19.7)   4300355.3 (33.6)
#>   bmi (%)                                                    <0.001
```

```
#>      Underweight          8844.4 ( 1.9)    359225.0 ( 2.8)
#>      healthy weight     173475.1 (36.8)   6277099.2 (49.0)
#>      Overweight         288617.9 (61.3)   6171091.9 (48.2)
#>   phyact (%)                                            <0.001
#>      Active              85140.3 (18.1)   3169053.4 (24.7)
#>      Inactive           274968.8 (58.4)   6355379.7 (49.6)
#>      Moderate           110828.4 (23.5)   3282983.0 (25.6)
#>   doctor = Yes (%)      445493.3 (94.6)  10732744.5 (83.8)  <0.001
#>   stress = stressed (%) 113282.5 (24.1)   3312383.7 (25.9)   0.023
#>   smoke (%)                                             <0.001
#>      Never smoker       119434.6 (25.4)   4244368.9 (33.1)
#>      Current smoker      97328.0 (20.7)   3397676.3 (26.5)
#>      Former smoker      254174.9 (54.0)   5165370.9 (40.3)
#>   drink (%)                                             <0.001
#>      Never drank         29444.3 ( 6.3)    715076.4 ( 5.6)
#>      Current drinker    344405.1 (73.1)  10841116.6 (84.6)
#>      Former driker       97088.1 (20.6)   1251223.1 ( 9.8)
#>   fruit (%)                                              0.001
#>      0-3 daily serving  111803.5 (23.7)   3338836.3 (26.1)
#>      4-6 daily serving  233403.5 (49.6)   6356044.7 (49.6)
#>      6+ daily serving   125730.4 (26.7)   3112535.1 (24.3)
#>   bp = Yes (%)          209257.0 (44.4)   1350561.0 (10.5)  <0.001
#>   diab = Yes (%)         78762.9 (16.7)    403654.4 ( 3.2)  <0.001
#>   province = North (%)     702.8 ( 0.1)     28246.6 ( 0.2)   0.005
#>   immigrate (%)                                         <0.001
#>      not immigrant      389553.0 (82.7)  10503766.2 (82.0)
#>      > 10 years          69008.0 (14.7)   1462544.4 (11.4)
#>      recent              12376.5 ( 2.6)    841105.5 ( 6.6)
```

How did they calculate the p-values? Hint: `svychisq` (see below).

Proportions and Design Effect

This part computes proportions and design effects, which help understand the influence of the sampling design on the estimated statistics.

```r
require(survey)
# Computing survey statistics on subsets of a survey defined by factor(s).
fit0a <- svyby(~CVD,~OA,design=w.design, svymean,deff=TRUE)
fit0a
#>               OA   CVDevent CVDno event  se.CVDevent se.CVDno event
#> Control Control 0.02791851   0.9720815 0.0005622009   0.0005622009
#> OA           OA 0.11481495   0.8851851 0.0032372797   0.0032372797
#>         DEff.CVDevent DEff.CVDno event
#> Control      1.939714         1.939714
#> OA           2.239419         2.239419
confint(fit0a)
#>                          2.5 %    97.5 %
#> Control:CVDevent    0.02681661 0.0290204
#> OA:CVDevent         0.10847000 0.1211599
#> Control:CVDno event 0.97097960 0.9731834
#> OA:CVDno event      0.87884010 0.8915300
# 7.45% OA patients estimated to have CVD event.
# 95% CI:  (0.067, 0.0816)
```

Let

- θ = parameter (population slope) and
- $\hat{(}\theta)$ = statistic (estimated slope).

$b = \frac{\sum[w(y_i - \bar{y})(x_i - \bar{x})]}{\sum[w(x_i - \bar{x})^2]}$

DE = Effect of complex survey on the SEs, relative to a SRS of equal size.

- $D^2(\hat{\theta}) = \frac{Var(\hat{\theta})_{ComplexSurvey}}{Var(\hat{\theta})_{SRS}}$
- $D^2(\hat{\theta}) = \frac{SE(\hat{\theta})^2_{ComplexSurvey}}{SE(\hat{\theta})^2_{SRS}}$

Note:

1. SE increases as value of weight increases (CCHS).
2. NHANES has more things to worry about (strata, PSU)

DEFF = 2 means that the variance of the sample proportion, when choosing the sample by complex survey sampling,

is nearly 2 times as large as the variance of the same estimator under simple random sampling/SRS.

```
fit0b <- svyby(~CVD,~diab,design=w.design, svymean,deff=TRUE)
fit0b
#>     diab   CVDevent CVDno event  se.CVDevent se.CVDno event DEff.CVDevent
#> No    No 0.03064837   0.9693516 0.0005536175   0.0005536175      1.853271
#> Yes  Yes 0.16326711   0.8367329 0.0064671407   0.0064671407      2.635026
#>     DEff.CVDno event
#> No          1.853271
#> Yes         2.635026
confint(fit0b)
#>                      2.5 %     97.5 %
#> No:CVDevent     0.0295633 0.03173344
#> Yes:CVDevent    0.1505917 0.17594247
#> No:CVDno event  0.9682666 0.97043670
#> Yes:CVDno event 0.8240575 0.84940825
```

Testing association

Here, Chi-square tests are conducted to test the association between different variables. Two variants of the test are used: Rao-Scott and Thomas-Rao modifications. These adaptations are used when the data come from a complex survey design.

- Tests for hypothesis
  - Rao-Scott modifications (chi-sq)
  - Thomas-Rao modifications (F)

```
# Rao-Scott modifications (chi-sq)
svychisq(~CVD+OA,design=w.design, statistic="Chisq")
#>
#>  Pearson's X^2: Rao & Scott adjustment
#>
#> data:  svychisq(~CVD + OA, design = w.design, statistic = "Chisq")
#> X-squared = 3249.7, df = 1, p-value < 2.2e-16

# Thomas-Rao modifications (F)
svychisq(~CVD+OA,design=w.design, statistic="F")
```

```
#>
#>  Pearson's X^2: Rao & Scott adjustment
#>
#> data:  svychisq(~CVD + OA, design = w.design, statistic = "F")
#> F = 1863, ndf = 1, ddf = 185612, p-value < 2.2e-16

# Both provide strong evidence to reject the null hypothesis.
# Conclusion: there is a significant (at 5%) association
# between CVD prevalence and OA.
svychisq(~CVD+fruit,design=w.design, statistic="F")
#>
#>  Pearson's X^2: Rao & Scott adjustment
#>
#> data:  svychisq(~CVD + fruit, design = w.design, statistic = "F")
#> F = 7.1241, ndf = 1.9758e+00, ddf = 3.6673e+05, p-value = 0.0008503
svychisq(~CVD+province,design=w.design, statistic="Chisq")
#>
#>  Pearson's X^2: Rao & Scott adjustment
#>
#> data:  svychisq(~CVD + province, design = w.design, statistic = "Chisq")
#> X-squared = 1.4848, df = 1, p-value = 0.00492
```

Saving data

Finally, the dataset, along with any new variables or subsets created during the analysis, is saved for future use.

```
save(w.design, analytic.miss, analytic2, file = "Data/surveydata/cchs123w.RData")
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

CCHS: Regression

This tutorial is for a complex data analysis, specifically using regression techniques to analyze survey data.

Let us load necessary R packages for the analysis:

```
# Load required packages
library(survey)
library(knitr)
library(car)
library(tableone)
library(DataExplorer)
library(Publish)
library(ROCR)
library(WeightedROC)
library(jtools)
library(MASS)
```

Load data

Loads a dataset and provides some quick data checks, like the dimensions and summary of weights.

```
load("Data/surveydata/cchs123w.RData")
ls()
#> [1] "analytic.miss"   "analytic2"       "has_annotations" "w.design"
dim(analytic.miss)
#> [1] 397173     23
dim(analytic2)
#> [1] 185613     22
summary(weights(w.design))
#> Length  Class   Mode
#>      0   NULL   NULL
```

Logistic for complex survey

Performs a simple logistic regression using the complex survey data, focusing on the relationship between cardiovascular disease and osteoarthritis.

```
formula0 <- as.formula(I(CVD=="event") ~ OA)

## Crude regression
fit2 <- svyglm(formula0,
              design = w.design,
              family = binomial(logit))
require(Publish)
publish(fit2)
#>  Variable   Units OddsRatio       CI.95 p-value
#>        OA Control       Ref
#>                OA      4.52 [4.19;4.87]  <1e-04
```

Multivariable analysis

Runs a more complex logistic regression model, adding multiple covariates to better understand the relationship.

```
formula1 <- as.formula(I(CVD=="event") ~ OA + age + sex + married + race +
              edu + income + bmi + phyact + doctor + stress +
              smoke + drink + fruit + bp + diab + province + immigrate)

fit3 <- svyglm(formula1,
              design = w.design,
              family = binomial(logit))
publish(fit3)
#>   Variable             Units OddsRatio         CI.95     p-value
#>         OA           Control       Ref
#>                           OA      1.52   [1.40;1.66]     < 1e-04
#>        age       20-29 years       Ref
#>                  30-39 years      1.29   [0.98;1.69]   0.0707636
#>                  40-49 years      2.74   [2.17;3.47]     < 1e-04
#>                  50-59 years      6.24   [4.97;7.83]     < 1e-04
#>                  60-64 years      9.71  [7.68;12.29]     < 1e-04
#>            65 years and over     15.85 [12.57;20.00]     < 1e-04
```

```
#>                          teen      0.46   [0.20;1.07]   0.0707295
#>        sex             Female       Ref
#>                          Male      1.73   [1.60;1.88]     < 1e-04
#>    married         not single       Ref
#>                        single      0.97   [0.90;1.05]   0.5209444
#>       race          Non-white       Ref
#>                         White      1.44   [1.19;1.75]   0.0002219
#>        edu           < 2ndary       Ref
#>                     2nd grad.      0.90   [0.80;1.00]   0.0512330
#>               Other 2nd grad.      0.97   [0.83;1.13]   0.6737665
#>                Post-2nd grad.      0.93   [0.85;1.02]   0.1016562
#>     income   $29,999 or less       Ref
#>              $30,000-$49,999      0.74   [0.67;0.81]     < 1e-04
#>              $50,000-$79,999      0.64   [0.58;0.72]     < 1e-04
#>              $80,000 or more      0.58   [0.51;0.66]     < 1e-04
#>        bmi       Underweight       Ref
#>                healthy weight      0.86   [0.67;1.10]   0.2350526
#>                    Overweight      0.88   [0.69;1.12]   0.3033213
#>     phyact             Active       Ref
#>                     Inactive      1.21   [1.10;1.34]   0.0001345
#>                     Moderate      1.08   [0.97;1.21]   0.1771985
#>     doctor                 No       Ref
#>                           Yes      1.75   [1.49;2.06]     < 1e-04
#>     stress  Not too stressed       Ref
#>                     stressed      1.30   [1.18;1.42]     < 1e-04
#>      smoke      Never smoker       Ref
#>                Current smoker      1.18   [1.05;1.32]   0.0050518
#>                 Former smoker      1.21   [1.11;1.33]     < 1e-04
#>      drink       Never drank       Ref
#>               Current drinker      0.82   [0.68;0.98]   0.0290605
#>                 Former driker      1.13   [0.93;1.36]   0.2133779
#>      fruit 0-3 daily serving       Ref
#>            4-6 daily serving      0.94   [0.86;1.03]   0.1758214
#>             6+ daily serving      1.09   [0.97;1.23]   0.1311029
#>         bp                 No       Ref
#>                           Yes      2.35   [2.16;2.55]     < 1e-04
#>       diab                 No       Ref
#>                           Yes      1.86   [1.66;2.07]     < 1e-04
#>   province              South       Ref
#>                         North      1.21   [0.90;1.62]   0.2103030
```

```
#>  immigrate     not immigrant       Ref
#>                   > 10 years      1.02   [0.89;1.16]   0.8057243
#>                       recent      1.06   [0.73;1.53]   0.7651069
```

Model fit assessment

Variability explained

Pseudo-R-square values indicate how much of the total variability in the outcomes is explainable by the fitted model (analogous to R-square). For a continuous outcome, we can compute R-square, while R-square cannot be calculated when the outcome variable is categorical, nominal or ordinal. We use pseudo-R-squared measures when the dependent variable is not continuous but a likelihood function is used to fit a model. Popular Pseudo-R-square measures are:

1. Cox/Snell (never reaches max 1)
2. Nagelkerke R-square (scaled to max 1)

- The larger Cox & Snell estimate is the better the model.
- These Pseudo-R-square values should be interpreted with caution (if not ignored).
- They offer little confidence in interpreting the model fit.
- Survey weighted version of them are available.
- Not trivial to decide which statistic to use under complex surveys.

Evaluates the model fit using Akaike Information Criterion (AIC) and pseudo R-squared metrics.

```
fit3 <- svyglm(formula1,
               design = w.design,
               family = quasibinomial(logit)) # publish does not work
AIC(fit3)
#>        eff.p          AIC     deltabar
#>    67.752064 45362.706032     2.053093

# AIC for survey weighted regressions
psrsq(fit3, method = "Cox-Snell")
```

```r
#> [1] 0.06091896
psrsq(fit3, method = "Nagelkerke")
#> [1] 0.2307586
# Nagelkerke and Cox-Snell pseudo-rsquared statistics
```

Backward Elimination

- Model comparisons
  - LRT-aprroximation
  - Wald-based

Checking one by one

Checks the significance of each variable one by one and removes those that are not statistically significant.

```r
round(sort(summary(fit3)$coef[,"Pr(>|t|)"]),2)
#>            (Intercept)   age65 years and over                  bpYes
#>                   0.00                   0.00                   0.00
#>         age60-64 years         age50-59 years                sexMale
#>                   0.00                   0.00                   0.00
#>                diabYes                   OAOA  income$80,000 or more
#>                   0.00                   0.00                   0.00
#>         age40-49 years  income$50,000-$79,999              doctorYes
#>                   0.00                   0.00                   0.00
#>  income$30,000-$49,999         stressstressed     smokeFormer smoker
#>                   0.00                   0.00                   0.00
#>         phyactInactive              raceWhite    smokeCurrent smoker
#>                   0.00                   0.00                   0.01
#>   drinkCurrent drinker           edu2nd grad.                ageteen
#>                   0.03                   0.05                   0.07
#>         age30-39 years      eduPost-2nd grad.  fruit6+ daily serving
#>                   0.07                   0.10                   0.13
#> fruit4-6 daily serving         phyactModerate          provinceNorth
#>                   0.18                   0.18                   0.21
#>     drinkFormer driker      bmihealthy weight          bmiOverweight
#>                   0.21                   0.24                   0.30
#>          marriedsingle     eduOther 2nd grad.        immigraterecent
```

```
#>                   0.52                   0.67                   0.77
#>    immigrate> 10 years
#>                   0.81
# bmiOverweight is associated with largest p-value
# but what about other categories?

regTermTest(fit3,~bmi) # coef of all bmi cat = 0
#> Wald test for bmi
#>  in svyglm(formula = formula1, design = w.design, family = quasibinomial(logit))
#> F =  0.7591291  on  2  and  185579  df: p= 0.46808
fit4 <- update(fit3, .~. -bmi)

anova(fit3, fit4)
#> Working (Rao-Scott+F) LRT for bmi
#>  in svyglm(formula = formula1, design = w.design, family = quasibinomial(logit))
#> Working 2logLR =  1.424634 p= 0.49071
#> (scale factors:  1.1 0.93 );  denominator df= 185579
# high p-value (in both wald and Anova) makes it more likely that you should exclude bmi
AIC(fit3,fit4)
#>         eff.p      AIC deltabar
#> [1,] 67.75206 45362.71 2.053093
#> [2,] 64.30460 45358.26 2.074342
round(sort(summary(fit4)$coef[,"Pr(>|t|)"]),2)
#>            (Intercept)   age65 years and over                   bpYes
#>                   0.00                   0.00                    0.00
#>         age60-64 years         age50-59 years                 sexMale
#>                   0.00                   0.00                    0.00
#>                diabYes                   OADA  income$80,000 or more
#>                   0.00                   0.00                    0.00
#>         age40-49 years  income$50,000-$79,999              doctorYes
#>                   0.00                   0.00                    0.00
#>  income$30,000-$49,999         stressstressed     smokeFormer smoker
#>                   0.00                   0.00                    0.00
#>         phyactInactive              raceWhite    smokeCurrent smoker
#>                   0.00                   0.00                    0.00
#>   drinkCurrent drinker           edu2nd grad.         age30-39 years
#>                   0.03                   0.05                    0.07
#>                ageteen      eduPost-2nd grad.  fruit6+ daily serving
#>                   0.07                   0.10                    0.13
#> fruit4-6 daily serving         phyactModerate          provinceNorth
```

```
#>                    0.17                    0.17                    0.21
#>     drinkFormer driker          marriedsingle     eduOther 2nd grad.
#>                    0.21                    0.53                    0.67
#>        immigraterecent    immigrate> 10 years
#>                    0.76                    0.81
```

Using AIC to automate

Uses stepwise regression guided by AIC to automatically select the most important variables.

```r
require(MASS)
formula1b <- as.formula(I(CVD=="event") ~ OA + age + sex)
fit1b <- svyglm(formula1b,
              design = w.design,
              family = binomial(logit))
fit5 <- stepAIC(fit1b, direction = "backward")
#> Start:  AIC=47384.51
#> I(CVD == "event") ~ OA + age + sex
#>
#>        Df Deviance   AIC
#> <none>       47353 47385
#> - sex   1    47634 47658
#> - OA    1    47679 47702
#> - age   6    54414 54291
```

```r
publish(fit5)
#>  Variable             Units OddsRatio         CI.95   p-value
#>        OA           Control       Ref
#>                          OA      1.81   [1.66;1.97]   < 1e-04
#>       age       20-29 years       Ref
#>                 30-39 years      1.40   [1.07;1.84]   0.01374
#>                 40-49 years      3.39   [2.69;4.27]   < 1e-04
#>                 50-59 years      9.42  [7.55;11.76]   < 1e-04
#>                 60-64 years     17.78 [14.22;22.23]   < 1e-04
#>           65 years and over     33.82 [27.26;41.97]   < 1e-04
#>                        teen      0.45   [0.20;1.02]   0.05462
#>       sex            Female       Ref
#>                        Male      1.57   [1.46;1.69]   < 1e-04
```

```r
round(sort(summary(fit5)$coef[,"Pr(>|t|)"]),2)
#>          (Intercept) age65 years and over       age60-64 years
#>                 0.00                 0.00                 0.00
#>       age50-59 years                 OAOA              sexMale
#>                 0.00                 0.00                 0.00
#>       age40-49 years       age30-39 years              ageteen
#>                 0.00                 0.01                 0.05
```

Using AIC, but keeping importants

Similar to the previous step but ensures certain important variables remain in the model.

```r
formula1c <- as.formula(I(CVD=="event") ~ OA + age + sex + married + race +
                        edu + income + bmi + phyact + fruit + bp + diab +
                        doctor + stress + smoke + drink + province + immigrate)
scope <- list(upper = ~ OA + age + sex + married + race +
                        edu + income + bmi + phyact + fruit + bp + diab +
                        doctor + stress + smoke + drink + province + immigrate,
              lower = ~ OA + age + sex + married + race +
                        edu + income + bmi + phyact + fruit + bp + diab)

fit1c <- svyglm(formula1c, design = w.design, family = binomial(logit))

fitstep <- step(fit1c, scope = scope, trace = FALSE, k = 2, direction = "backward")
# k = 2 gives the genuine AIC
```

```r
publish(fitstep)
#>  Variable             Units OddsRatio         CI.95     p-value
#>        OA           Control       Ref
#>                          OA      1.52   [1.40;1.66]     < 1e-04
#>       age       20-29 years       Ref
#>                 30-39 years      1.29   [0.98;1.70]   0.0697699
#>                 40-49 years      2.74   [2.17;3.47]     < 1e-04
#>                 50-59 years      6.24   [4.97;7.83]     < 1e-04
#>                 60-64 years      9.71  [7.68;12.28]     < 1e-04
#>           65 years and over     15.85 [12.57;19.98]     < 1e-04
#>                        teen      0.46   [0.20;1.07]   0.0705660
#>       sex            Female       Ref
```

```
#>                          Male      1.73   [1.60;1.88]     < 1e-04
#>   married         not single       Ref
#>                       single      0.97   [0.90;1.05]   0.5042496
#>      race          Non-white       Ref
#>                        White      1.41   [1.18;1.70]   0.0001986
#>       edu           < 2ndary       Ref
#>                    2nd grad.      0.90   [0.80;1.00]   0.0524940
#>             Other 2nd grad.      0.97   [0.83;1.13]   0.6732446
#>              Post-2nd grad.      0.93   [0.85;1.02]   0.1045975
#>    income   $29,999 or less       Ref
#>             $30,000-$49,999      0.74   [0.67;0.81]     < 1e-04
#>             $50,000-$79,999      0.64   [0.58;0.72]     < 1e-04
#>             $80,000 or more      0.58   [0.51;0.66]     < 1e-04
#>       bmi        Underweight       Ref
#>              healthy weight      0.86   [0.67;1.10]   0.2316915
#>                  Overweight      0.88   [0.69;1.12]   0.2982852
#>    phyact             Active       Ref
#>                    Inactive      1.22   [1.10;1.34]   0.0001227
#>                    Moderate      1.08   [0.97;1.21]   0.1754422
#>     fruit 0-3 daily serving       Ref
#>           4-6 daily serving      0.94   [0.86;1.03]   0.1807666
#>            6+ daily serving      1.09   [0.97;1.23]   0.1295281
#>        bp                 No       Ref
#>                          Yes      2.35   [2.16;2.55]     < 1e-04
#>      diab                 No       Ref
#>                          Yes      1.85   [1.66;2.07]     < 1e-04
#>    doctor                 No       Ref
#>                          Yes      1.75   [1.49;2.05]     < 1e-04
#>    stress  Not too stressed       Ref
#>                    stressed      1.30   [1.18;1.42]     < 1e-04
#>     smoke      Never smoker       Ref
#>              Current smoker      1.17   [1.05;1.31]   0.0053412
#>               Former smoker      1.21   [1.10;1.33]     < 1e-04
#>     drink       Never drank       Ref
#>             Current drinker      0.82   [0.68;0.98]   0.0254942
#>               Former driker      1.12   [0.93;1.36]   0.2205315
round(sort(summary(fitstep)$coef[,"Pr(>|t|)"]),2)
#>            (Intercept)   age65 years and over                 bpYes
#>                   0.00                   0.00                  0.00
#>         age60-64 years         age50-59 years               sexMale
```

```
#>                   0.00                   0.00                   0.00
#>                diabYes                   OAOA  income$80,000 or more
#>                   0.00                   0.00                   0.00
#>         age40-49 years  income$50,000-$79,999              doctorYes
#>                   0.00                   0.00                   0.00
#>  income$30,000-$49,999         stressstressed     smokeFormer smoker
#>                   0.00                   0.00                   0.00
#>         phyactInactive              raceWhite    smokeCurrent smoker
#>                   0.00                   0.00                   0.01
#>   drinkCurrent drinker           edu2nd grad.         age30-39 years
#>                   0.03                   0.05                   0.07
#>                ageteen      eduPost-2nd grad.  fruit6+ daily serving
#>                   0.07                   0.10                   0.13
#>         phyactModerate fruit4-6 daily serving     drinkFormer driker
#>                   0.18                   0.18                   0.22
#>      bmihealthy weight          bmiOverweight          marriedsingle
#>                   0.23                   0.30                   0.50
#>     eduOther 2nd grad.
#>                   0.67
```

Assess interactions

Check biologically interesting ones.

0.0.0.0.1 * Check one by one

Checks if there is a significant interaction effect between 'age' and 'sex'.

```
fit8a <- update(fitstep, .~. + interaction(age,sex))
anova(fitstep, fit8a) # keep interaction
#> Working (Rao-Scott+F) LRT for interaction(age, sex)
#>  in svyglm(formula = I(CVD == "event") ~ OA + age + sex + married +
#>     race + edu + income + bmi + phyact + fruit + bp + diab +
#>     doctor + stress + smoke + drink + interaction(age, sex),
#>     design = w.design, family = binomial(logit))
#> Working 2logLR =  40.16528 p= 1.2167e-06
#> (scale factors:  1.3 1.2 1.2 0.93 0.78 0.71 );  denominator df= 185576
```

Checks if there is a significant interaction effect between 'sex' and 'diabetes'.

```
fit8b <- update(fitstep, .~. + interaction(sex,diab))
anova(fitstep, fit8b) # Do not keep this interaction
#> Working (Rao-Scott+F) LRT for interaction(sex, diab)
#>  in svyglm(formula = I(CVD == "event") ~ OA + age + sex + married +
#>     race + edu + income + bmi + phyact + fruit + bp + diab +
#>     doctor + stress + smoke + drink + interaction(sex, diab),
#>     design = w.design, family = binomial(logit))
#> Working 2logLR =  0.4591456 p= 0.49597
#> df=1;  denominator df= 185581
```

Checks if there is a significant interaction effect between 'BMI' and 'diabetes'.

```
fit8c <- update(fitstep, .~. + interaction(bmi,diab))
anova(fitstep, fit8c) # keep this interaction
#> Working (Rao-Scott+F) LRT for interaction(bmi, diab)
#>  in svyglm(formula = I(CVD == "event") ~ OA + age + sex + married +
#>     race + edu + income + bmi + phyact + fruit + bp + diab +
#>     doctor + stress + smoke + drink + interaction(bmi, diab),
#>     design = w.design, family = binomial(logit))
#> Working 2logLR =  7.92727 p= 0.02533
#> (scale factors:  1.4 0.6 );  denominator df= 185580
```

0.0.0.0.2 * Add all significant interactions in 1 model

Updates the model to include significant interaction terms.

Note that we have 0 effect modifier, 2 interactions

```
fit9 <- update(fitstep, .~. + interaction(age,sex) + interaction(bmi,diab))
require(jtools)
summ(fit9, confint = TRUE, digits = 3)
```

```
fit9 <- update(fitstep, .~. + age:sex + bmi:diab)
publish(fit9)
#>                                          Variable              Units OddsRatio          CI.
```

Observations	185613
Dependent variable	I(CVD == "event")
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R² (Cragg-Uhler)	0.233
Pseudo-R² (McFadden)	0.207
AIC	41548.160

```
#>                                          OA          Control      Ref
#>                                                          OA     1.55   [1.42;1.6
#>                                     married      not single      Ref
#>                                                      single     0.98   [0.91;1.0
#>                                        race       Non-white      Ref
#>                                                       White     1.42   [1.18;1.7
#>                                         edu        < 2ndary      Ref
#>                                                   2nd grad.     0.90   [0.81;1.0
#>                                                Other 2nd grad.  0.96   [0.83;1.1
#>                                                 Post-2nd grad.  0.92   [0.84;1.0
#>                                      income  $29,999 or less     Ref
#>                                              $30,000-$49,999    0.74   [0.67;0.8
#>                                              $50,000-$79,999    0.64   [0.58;0.7
#>                                              $80,000 or more    0.57   [0.50;0.6
#>                                     phyact           Active      Ref
#>                                                    Inactive     1.21   [1.10;1.3
#>                                                    Moderate     1.08   [0.97;1.2
#>                                       fruit 0-3 daily serving   Ref
#>                                             4-6 daily serving   0.94   [0.86;1.0
#>                                              6+ daily serving   1.10   [0.98;1.2
#>                                         bp               No      Ref
#>                                                         Yes     2.37   [2.18;2.5
#>                                     doctor               No      Ref
#>                                                         Yes     1.71   [1.45;2.0
#>                                     stress  Not too stressed     Ref
#>                                                    stressed     1.29   [1.18;1.4
#>                                      smoke     Never smoker      Ref
#>                                              Current smoker     1.16   [1.04;1.3
#>                                               Former smoker     1.19   [1.09;1.3
```

```
#>                                                 drink        Never drank        Ref
#>                                                          Current drinker       0.81   [0.68;0.9
#>                                                            Former driker       1.12   [0.93;1.3
#>                 age(20-29 years): sex(Male vs Female)                          0.97   [0.64;1.4
#>                 age(30-39 years): sex(Male vs Female)                          1.06   [0.75;1.4
#>                 age(40-49 years): sex(Male vs Female)                          1.32   [1.07;1.6
#>                 age(50-59 years): sex(Male vs Female)                          2.18   [1.85;2.5
#>                 age(60-64 years): sex(Male vs Female)                          2.14   [1.80;2.5
#>           age(65 years and over): sex(Male vs Female)                          1.78   [1.58;1.9
#>                       age(teen): sex(Male vs Female)                           1.11   [0.22;5.6
#>          sex(Female): age(30-39 years vs 20-29 years)                          1.24   [0.87;1.7
#>          sex(Female): age(40-49 years vs 20-29 years)                          2.35   [1.75;3.1
#>          sex(Female): age(50-59 years vs 20-29 years)                          3.94   [2.95;5.2
#>          sex(Female): age(60-64 years vs 20-29 years)                          6.25   [4.66;8.3
#>    sex(Female): age(65 years and over vs 20-29 years)                         11.37  [8.60;15.0
#>                 sex(Female): age(teen vs 20-29 years)                          0.44   [0.11;1.6
#>            sex(Male): age(30-39 years vs 20-29 years)                          1.35   [0.89;2.0
#>            sex(Male): age(40-49 years vs 20-29 years)                          3.18   [2.21;4.5
#>            sex(Male): age(50-59 years vs 20-29 years)                          8.85  [6.24;12.5
#>            sex(Male): age(60-64 years vs 20-29 years)                         13.73  [9.61;19.6
#>      sex(Male): age(65 years and over vs 20-29 years)                         20.77 [14.59;29.5
#>                   sex(Male): age(teen vs 20-29 years)                          0.50   [0.18;1.3
#>                     bmi(Underweight): diab(Yes vs No)                          5.72  [1.59;20.6
#>                  bmi(healthy weight): diab(Yes vs No)                          1.44   [1.19;1.7
#>                      bmi(Overweight): diab(Yes vs No)                          1.93   [1.70;2.2
#>          diab(No): bmi(healthy weight vs Underweight)                          0.97   [0.76;1.2
#>              diab(No): bmi(Overweight vs Underweight)                          0.97   [0.76;1.2
#>         diab(Yes): bmi(healthy weight vs Underweight)                          0.25   [0.07;0.8
#>             diab(Yes): bmi(Overweight vs Underweight)                          0.33   [0.09;1.1
```

```
basic.model <- eval(fit5$call[[2]])
basic.model
#> I(CVD == "event") ~ OA + age + sex
#> attr(,"variables")
#> list(I(CVD == "event"), OA, age, sex)
#> attr(,"factors")
#>                    OA age sex
#> I(CVD == "event")  0   0   0
#> OA                 1   0   0
#> age                0   1   0
```

```
#> sex                 0   0   1
#> attr(,"term.labels")
#> [1] "OA"  "age" "sex"
#> attr(,"order")
#> [1] 1 1 1
#> attr(,"intercept")
#> [1] 1
#> attr(,"response")
#> [1] 1
#> attr(,".Environment")
#> <environment: R_GlobalEnv>
#> attr(,"predvars")
#> list(I(CVD == "event"), OA, age, sex)
#> attr(,"dataClasses")
#> I(CVD == "event")                OA               age               sex
#>         "logical"          "factor"          "factor"          "factor"
#>         (weights)
#>         "numeric"

aic.int.model <- eval(fit9$call[[2]])
aic.int.model
#> I(CVD == "event") ~ OA + age + sex + married + race + edu + income +
#>     bmi + phyact + fruit + bp + diab + doctor + stress + smoke +
#>     drink + age:sex + bmi:diab
```

Saving data

Saves the final regression models for future use.

```
save(basic.model, aic.int.model, file = "Data/surveydata/cchs123w2.RData")
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

	Est.	2.5%	97.5%	t val.	p
(Intercept)	-5.590	-6.046	-5.135	-24.069	0.000
OAOA	0.438	0.352	0.523	10.043	0.000
age30-39 years	0.299	-0.118	0.717	1.405	0.160
age40-49 years	1.158	0.794	1.522	6.236	0.000
age50-59 years	2.181	1.831	2.531	12.212	0.000
age60-64 years	2.619	2.263	2.976	14.395	0.000
age65 years and over	3.033	2.680	3.387	16.833	0.000
ageteen	-0.695	-1.704	0.314	-1.350	0.177
sexMale	-0.028	-0.447	0.390	-0.133	0.895
marriedsingle	-0.016	-0.096	0.064	-0.382	0.703
raceWhite	0.348	0.165	0.531	3.724	0.000
edu2nd grad.	-0.107	-0.217	0.003	-1.906	0.057
eduOther 2nd grad.	-0.039	-0.192	0.114	-0.498	0.618
eduPost-2nd grad.	-0.081	-0.173	0.010	-1.746	0.081
income$30,000-$49,999	-0.304	-0.400	-0.208	-6.215	0.000
income$50,000-$79,999	-0.444	-0.552	-0.336	-8.034	0.000
income$80,000 or more	-0.555	-0.684	-0.426	-8.424	0.000
bmihealthy weight	-1.406	-2.677	-0.135	-2.167	0.030
bmiOverweight	-1.110	-2.375	0.154	-1.721	0.085
phyactInactive	0.194	0.094	0.293	3.817	0.000
phyactModerate	0.077	-0.034	0.187	1.353	0.176
fruit4-6 daily serving	-0.062	-0.155	0.031	-1.313	0.189
fruit6+ daily serving	0.094	-0.023	0.211	1.579	0.114
bpYes	0.861	0.778	0.944	20.354	0.000
diabYes	1.745	0.462	3.027	2.667	0.008
doctorYes	0.535	0.372	0.697	6.447	0.000
stressstressed	0.256	0.164	0.347	5.483	0.000
smokeCurrent smoker	0.152	0.039	0.266	2.628	0.009
smokeFormer smoker	0.177	0.082	0.272	3.654	0.000
drinkCurrent drinker	-0.205	-0.381	-0.029	-2.277	0.023
drinkFormer driker	0.116	-0.072	0.303	1.210	0.226
interaction(age, sex)30-39 years.Female	-0.083	-0.622	0.457	-0.300	0.764
interaction(age, sex)40-49 years.Female	-0.302	-0.766	0.161	-1.278	0.201
interaction(age, sex)50-59 years.Female	-0.810	-1.258	-0.362	-3.543	0.000
interaction(age, sex)60-64 years.Female	-0.787	-1.237	-0.337	-3.428	0.001
interaction(age, sex)65 years and over.Female	-0.603	-1.035	-0.170	-2.733	0.006
interaction(age, sex)teen.Female	-0.129	-1.806	1.548	-0.151	0.880
interaction(bmi, diab)healthy weight.No	1.380	0.085	2.675	2.089	0.037
interaction(bmi, diab)Overweight.No	1.085	-0.204	2.373	1.650	0.099

Standard errors: Robust

CCHS: Performance

The tutorial outlines the process for evaluating the performance of logistic regression models fitted to complex survey data using R. It focuses on two major aspects: creating Receiver Operating Characteristic (ROC) curves and conducting Archer and Lemeshow Goodness of Fit tests. Here AUC is a measure to evaluate the predictive accuracy of the model, and Archer and Lemeshow test is a statistical test to evaluate how well your model fits the observed data.

We start by importing the required R packages.

```
# Load required packages
library(survey)
library(ROCR)
library(WeightedROC)
```

Load data

It loads two datasets from the specified paths.

```
load("Data/surveydata/cchs123w.RData")
load("Data/surveydata/cchs123w2.RData")
ls()
#> [1] "aic.int.model"   "analytic.miss"   "analytic2"       "basic.model"
#> [5] "has_annotations" "w.design"
dim(analytic.miss)
#> [1] 397173     23
dim(analytic2)
#> [1] 185613     22
```

Three different logistic regression models are fitted to the data:

- Simple model: Model with only OA as a predictor
- Basic model: Model with OA, age and sex
- Complex model: Model with many predictors and some interaction terms

```r
# Formula for Simple model
simple.model <- as.formula(I(CVD=="event") ~ OA)
simple.model
#> I(CVD == "event") ~ OA

# Formula for Basic model
basic.model
#> I(CVD == "event") ~ OA + age + sex
#> attr(,"variables")
#> list(I(CVD == "event"), OA, age, sex)
#> attr(,"factors")
#>                    OA age sex
#> I(CVD == "event")  0   0   0
#> OA                 1   0   0
#> age                0   1   0
#> sex                0   0   1
#> attr(,"term.labels")
#> [1] "OA"  "age" "sex"
#> attr(,"order")
#> [1] 1 1 1
#> attr(,"intercept")
#> [1] 1
#> attr(,"response")
#> [1] 1
#> attr(,".Environment")
#> <environment: R_GlobalEnv>
#> attr(,"predvars")
#> list(I(CVD == "event"), OA, age, sex)
#> attr(,"dataClasses")
#> I(CVD == "event")                OA               age               sex
#>         "logical"          "factor"          "factor"          "factor"
#>         (weights)
#>         "numeric"

# Formula for the Complex model with interactions
aic.int.model
```

```
#> I(CVD == "event") ~ OA + age + sex + married + race + edu + income +
#>     bmi + phyact + fruit + bp + diab + doctor + stress + smoke +
#>     drink + age:sex + bmi:diab
```

```r
library(survey)

# Simple model
fit0 <- svyglm(simple.model,
               design = w.design,
               family = binomial(logit))
#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!

# Basic model
fit5 <- svyglm(basic.model,
               design = w.design,
               family = binomial(logit))
#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!

# Complex model with interactions
fit9 <- svyglm(aic.int.model,
               design = w.design,
               family = binomial(logit))
#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

Model performance

ROC curve

This section defines a function, `svyROCw`, to plot the ROC curves and calculate the area under the curve (AUC). The function can handle both weighted and unweighted survey data.

- The appropriateness of the fitted logistic regression model needs to be examined before it is accepted for use.
- Plotting the pairs of - sensitivities vs - 1-specificities on a scatter plot provides a Receiver Operating Characteristic (ROC) curve.
- The area under the ROC curve = AUC / C-statistic.
- ROC/AUC should consider weights for complex surveys.

Grading Guidelines for AUC values:

- 0.90-1.0 excellent discrimination (unusual)
- 0.80-0.90 good discrimination
- 0.70-0.80 fair discrimination
- 0.60-0.70 poor discrimination
- 0.50-0.60 failed discrimination

```r
require(ROCR)
# WeightedROC may not be on cran for all R versions
# devtools::install_github("tdhock/WeightedROC")

library(WeightedROC)
svyROCw <- function(fit=fit,outcome=analytic2$CVD=="event", weight = NULL){
  # ROC curve for
  # Survey Data with Logistic Regression
  if (is.null(weight)){ # require(ROCR)
    prob <- predict(fit, type = "response")
  pred <- prediction(as.vector(prob), outcome)
  perf <- performance(pred, "tpr", "fpr")
  auc <- performance(pred, measure = "auc")
  auc <- auc@y.values[[1]]
  roc.data <- data.frame(fpr = unlist(perf@x.values), tpr = unlist(perf@y.values),
      model = "Logistic")
  with(data = roc.data,plot(fpr, tpr, type="l", xlim=c(0,1), ylim=c(0,1), lwd=1,
     xlab="1 - specificity", ylab="Sensitivity",
     main = paste("AUC = ", round(auc,3))))
  mtext("Unweighted ROC")
  abline(0,1, lty=2)
  } else { # library(WeightedROC)
    outcome <- as.numeric(outcome)
  pred <- predict(fit, type = "response")
  tp.fp <- WeightedROC(pred, outcome, weight)
  auc <- WeightedAUC(tp.fp)
  with(data = tp.fp,plot(FPR, TPR, type="l", xlim=c(0,1), ylim=c(0,1), lwd=1,
     xlab="1 - specificity", ylab="Sensitivity",
     main = paste("AUC = ", round(auc,3))))
  abline(0,1, lty=2)
  mtext("Weighted ROC")
  }
}
```

```r
summary(analytic2$weight)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    1.17   71.56  137.95  214.61  261.91 7154.95
analytic2$corrected.weight <- weights(w.design)
summary(analytic2$corrected.weight)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    0.39   23.85   45.98   71.54   87.30 2384.98
svyROCw(fit=fit0,outcome=analytic2$CVD=="event", weight = analytic2$corrected.weight)
```

```r
svyROCw(fit=fit5,outcome=analytic2$CVD=="event", weight = analytic2$corrected.weight)
```

```
svyROCw(fit=fit9,outcome=analytic2$CVD=="event", weight = analytic2$corrected.weight)
```

```
# This function does not take in to account of strata/cluster
```

Archer and Lemeshow test

This test helps to evaluate how well the model fits the data. A Goodness of Fit (GOF) function `AL.gof` is defined. If the

p-value from this test is greater than a certain threshold (e.g., 0.05), the model fit is considered acceptable.

- Hosmer Lemeshow-type tests are most useful as a very crude way to screen for fit problems, and should not be taken as a definitive diagnostic of a 'good' fit.
  - problem in small sample size
  - Dependent on G (group)
- Archer and Lemeshow (2006) extended the standard Hosmer and Lemeshow GOF test for complex surveys.
- After fitting the survey weighted logistic regression, the F-adjusted mean residual goodness-of-fit test could suggest
  - no evidence of lack of fit (if P-value > a reasonable cut-point, e.g., 0.05)
  - evidence of lack of fit (if P-value < a reasonable cut-point, e.g., 0.05)

```
AL.gof <- function(fit=fit, data = analytic2,
                   weight = "corrected.weight"){
  # Archer-Lemeshow Goodness of Fit Test for
  # Survey Data with Logistic Regression
  r <- residuals(fit, type="response")
  f<-fitted(fit)
  breaks.g <- c(-Inf, quantile(f,  (1:9)/10), Inf)
  breaks.g <- breaks.g + seq_along(breaks.g) * .Machine$double.eps
  g<- cut(f, breaks.g)
  data2g <- cbind(data,r,g)
  newdesign <- svydesign(id=~1,
                         weights=as.formula(paste0("~",weight)),
                        data=data2g)
  decilemodel<- svyglm(r~g, design=newdesign)
  res <- regTermTest(decilemodel, ~g)
  return(res)
}
```

```
AL.gof(fit0, analytic2, weight ="corrected.weight")
#> Wald test for g
#>  in svyglm(formula = r ~ g, design = newdesign)
#> F =  2.819403e-22  on  1  and  185611  df: p= 1
```

```r
AL.gof(fit5, analytic2, weight ="corrected.weight")
#> Wald test for g
#>  in svyglm(formula = r ~ g, design = newdesign)
#> F =  2.795204  on  8  and  185604  df: p= 0.0042898
AL.gof(fit9, analytic2, weight = "corrected.weight")
#> Wald test for g
#>  in svyglm(formula = r ~ g, design = newdesign)
#> F =  2.650332  on  9  and  185603  df: p= 0.0045417
```

Additional function

If the survey data contains strata and cluster, then the following function will be useful:

```r
AL.gof2 <- function(fit=fit7, data = analytic,
                    weight = "corrected.weight", psu = "psu", strata= "strata"){
  # Archer-Lemeshow Goodness of Fit Test for
  # Survey Data with Logistic Regression
  r <- residuals(fit, type="response")
  f<-fitted(fit)
  breaks.g <- c(-Inf, quantile(f,  (1:9)/10), Inf)
  breaks.g <- breaks.g + seq_along(breaks.g) * .Machine$double.eps
  g<- cut(f, breaks.g)
  data2g <- cbind(data,r,g)
  newdesign <- svydesign(id=as.formula(paste0("~",psu)),
                         strata=as.formula(paste0("~",strata)),
                         weights=as.formula(paste0("~",weight)),
                        data=data2g, nest = TRUE)
  decilemodel<- svyglm(r~g, design=newdesign)
  res <- regTermTest(decilemodel, ~g)
  return(res)
}
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

NHANES: Blood Pressure

The tutorial aims to guide the user through the process of analyzing health survey data, focusing specifically on the relationship between various demographic factors and blood pressure levels.

Required packages are imported for data manipulation and statistical analysis.

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

Load data

NHANES survey data is loaded into the workspace.

```
load(file = "Data/surveydata/NHANESsample.Rdata")
ls()
#> [1] "analytic.data"   "has_annotations"
```

Regression summary

Bivariate Regression

Four separate models are constructed to explore the relationship between blood pressure and individual demographic variables like race, age, gender, and marital status.

```
summary(fit1r <- glm(blood.pressure ~ race, data=analytic.data))
#>
#> Call:
#> glm(formula = blood.pressure ~ race, data = analytic.data)
```

```
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -66.501   -8.437    0.217    8.400   53.563
#>
#> Coefficients:
#>                        Estimate Std. Error t value Pr(>|t|)
#> (Intercept)             65.4408     0.3566 183.528  < 2e-16 ***
#> raceNon-Hispanic Black   2.1594     0.4891   4.415 1.02e-05 ***
#> raceNon-Hispanic White   0.9957     0.4498   2.214   0.0269 *
#> raceOther Hispanic       0.3422     0.5553   0.616   0.5378
#> raceOther race           3.0601     0.5326   5.746 9.54e-09 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 168.7193)
#>
#>     Null deviance: 1202625  on 7086  degrees of freedom
#> Residual deviance: 1194870  on 7082  degrees of freedom
#>   (2884 observations deleted due to missingness)
#> AIC: 56463
#>
#> Number of Fisher Scoring iterations: 2
```

```
summary(fit1a <- glm(blood.pressure ~ age.centred, data=analytic.data))
#>
#> Call:
#> glm(formula = blood.pressure ~ age.centred, data = analytic.data)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -59.275   -8.046    0.366    8.083   54.057
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 68.297917   0.158325   431.4   <2e-16 ***
#> age.centred  0.179501   0.006623    27.1   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
```

```
#> (Dispersion parameter for gaussian family taken to be 153.7967)
#>
#>     Null deviance: 1202625  on 7086  degrees of freedom
#> Residual deviance: 1089649  on 7085  degrees of freedom
#>   (2884 observations deleted due to missingness)
#> AIC: 55804
#>
#> Number of Fisher Scoring iterations: 2
```

```
summary(fit1g <- glm(blood.pressure ~ gender, data=analytic.data))
#>
#> Call:
#> glm(formula = blood.pressure ~ gender, data = analytic.data)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -63.998   -7.998    0.002    8.514   54.002
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   67.4863     0.2210 305.409  < 2e-16 ***
#> genderFemale  -1.4885     0.3091  -4.816  1.5e-06 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 169.1886)
#>
#>     Null deviance: 1202625  on 7086  degrees of freedom
#> Residual deviance: 1198702  on 7085  degrees of freedom
#>   (2884 observations deleted due to missingness)
#> AIC: 56480
#>
#> Number of Fisher Scoring iterations: 2
```

```
summary(fit1m <- glm(blood.pressure ~ marital, data=analytic.data))
#>
#> Call:
#> glm(formula = blood.pressure ~ marital, data = analytic.data)
#>
#> Deviance Residuals:
```

```
#>     Min       1Q   Median       3Q      Max
#> -55.664   -7.625   -0.593    7.407   50.336
#>
#> Coefficients:
#>                           Estimate Std. Error t value Pr(>|t|)
#> (Intercept)                70.5934     0.2129 331.648   <2e-16 ***
#> maritalNever married       -0.9293     0.4430  -2.098   0.0360 *
#> maritalPreviously married  -0.9689     0.4200  -2.307   0.0211 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 141.0891)
#>
#>     Null deviance: 723763  on 5124  degrees of freedom
#> Residual deviance: 722658  on 5122  degrees of freedom
#>   (4846 observations deleted due to missingness)
#> AIC: 39915
#>
#> Number of Fisher Scoring iterations: 2
```

Multivariate Regression

A more comprehensive model is built, combining all the aforementioned demographic factors to predict blood pressure.

```
summary(fit1 <- glm(blood.pressure ~ race + age.centred + gender + marital, data=analytic.data)
#>
#> Call:
#> glm(formula = blood.pressure ~ race + age.centred + gender +
#>     marital, data = analytic.data)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -58.794   -7.601   -0.159    7.409   52.141
#>
#> Coefficients:
#>                            Estimate Std. Error t value Pr(>|t|)
#> (Intercept)               70.879238   0.436140 162.515  < 2e-16 ***
#> raceNon-Hispanic Black     2.632298   0.539123   4.883 1.08e-06 ***
```

```
#> raceNon-Hispanic White     0.432023   0.486314   0.888 0.374389
#> raceOther Hispanic         0.042382   0.596004   0.071 0.943313
#> raceOther race             2.854707   0.576021   4.956 7.43e-07 ***
#> age.centred               -0.039408   0.010467  -3.765 0.000168 ***
#> genderFemale              -2.728473   0.332506  -8.206 2.87e-16 ***
#> maritalNever married      -1.786003   0.468287  -3.814 0.000138 ***
#> maritalPreviously married  0.001345   0.439870   0.003 0.997560
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 137.6355)
#>
#>     Null deviance: 723763  on 5124  degrees of freedom
#> Residual deviance: 704143  on 5116  degrees of freedom
#>   (4846 observations deleted due to missingness)
#> AIC: 39794
#>
#> Number of Fisher Scoring iterations: 2
```

Including survey features

The tutorial takes into account the complex survey design by creating a new survey design object (based on survey features such as, sampling weight, strata and cluster).

```
require(survey)
#> Loading required package: survey
#> Loading required package: grid
#> Loading required package: Matrix
#> Loading required package: survival
#>
#> Attaching package: 'survey'
#> The following object is masked from 'package:graphics':
#>
#>     dotchart
analytic.design <- svydesign(strata=~SDMVSTRA, id=~SDMVPSU, weights=~WTMEC2YR, data=analytic.da
```

Box plots and summary statistics are generated to visualize how blood pressure varies across different demographic groups,

considering the survey weights and design.

```
svyboxplot(blood.pressure~race, design = analytic.design, col="grey",
           ylab="Blood Pressure",
           xlab ="Race")
```

```
svyby(~blood.pressure, by=~race, design = analytic.design, na.rm=TRUE, ci=TRUE, svymean)
#>                                  race blood.pressure        se
#> Mexican American     Mexican American       66.61500 0.2756435
#> Non-Hispanic Black Non-Hispanic Black       68.66411 0.6794312
#> Non-Hispanic White Non-Hispanic White       68.53319 0.4619354
#> Other Hispanic         Other Hispanic       66.81714 0.8128909
#> Other race                 Other race       69.22413 0.4280046
svyboxplot(blood.pressure~gender, design = analytic.design, col="grey",
           ylab="Blood Pressure",
           xlab ="Gender")
```

```
svyby (~blood.pressure, by=~gender, design = analytic.design, na.rm=TRUE, ci=TRUE, svymean)
#>         gender blood.pressure        se
#> Male     Male       69.28980 0.4552303
#> Female Female       67.38015 0.4446360
svyboxplot(blood.pressure~marital, design = analytic.design, col="grey",
           ylab="Blood Pressure",
           xlab ="Marital Status")
```

```r
svyby (~blood.pressure, by=~marital, design = analytic.design, na.rm=TRUE, ci=TRUE, svymean)
#>                              marital blood.pressure        se
#> Married                      Married       70.82622 0.4978471
#> Never married          Never married       69.62996 0.5592240
#> Previously married Previously married       70.32240 0.5233053
```

Bivariate Regression

New regression models are constructed using the survey design object to understand how each demographic variable impacts blood pressure while taking survey features into consideration.

```r
require(survey)
summary(fit1r <- svyglm(blood.pressure ~ race, design=analytic.design))
#>
#> Call:
#> svyglm(formula = blood.pressure ~ race, design = analytic.design)
#>
#> Survey design:
#> svydesign(strata = ~SDMVSTRA, id = ~SDMVPSU, weights = ~WTMEC2YR,
#>     data = analytic.data, nest = TRUE)
#>
#> Coefficients:
#>                        Estimate Std. Error t value Pr(>|t|)
#> (Intercept)             66.6150     0.2756 241.671  < 2e-16 ***
#> raceNon-Hispanic Black   2.0491     0.6144   3.335 0.006650 **
#> raceNon-Hispanic White   1.9182     0.4269   4.493 0.000912 ***
#> raceOther Hispanic       0.2021     0.7825   0.258 0.800928
#> raceOther race           2.6091     0.3998   6.527 4.27e-05 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 179.8689)
#>
#> Number of Fisher Scoring iterations: 2
```

```r
summary(fit1a <- svyglm(blood.pressure ~ age.centred, design=analytic.design))
#>
```

```
#> Call:
#> svyglm(formula = blood.pressure ~ age.centred, design = analytic.design)
#>
#> Survey design:
#> svydesign(strata = ~SDMVSTRA, id = ~SDMVPSU, weights = ~WTMEC2YR,
#>     data = analytic.data, nest = TRUE)
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 69.25489    0.47091  147.07  < 2e-16 ***
#> age.centred  0.15142    0.01503   10.08  8.5e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 169.12)
#>
#> Number of Fisher Scoring iterations: 2
```

```r
summary(fit1g <- svyglm(blood.pressure ~ gender, design=analytic.design))
#>
#> Call:
#> svyglm(formula = blood.pressure ~ gender, design = analytic.design)
#>
#> Survey design:
#> svydesign(strata = ~SDMVSTRA, id = ~SDMVPSU, weights = ~WTMEC2YR,
#>     data = analytic.data, nest = TRUE)
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   69.2898     0.4552 152.208  < 2e-16 ***
#> genderFemale  -1.9096     0.4233  -4.511 0.000488 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 179.4491)
#>
#> Number of Fisher Scoring iterations: 2
```

```r
summary(fit1m <- svyglm(blood.pressure ~ marital, design=analytic.design))
#>
#> Call:
#> svyglm(formula = blood.pressure ~ marital, design = analytic.design)
#>
#> Survey design:
#> svydesign(strata = ~SDMVSTRA, id = ~SDMVPSU, weights = ~WTMEC2YR,
#>     data = analytic.data, nest = TRUE)
#>
#> Coefficients:
#>                           Estimate Std. Error t value Pr(>|t|)
#> (Intercept)                70.8262     0.4978 142.265   <2e-16 ***
#> maritalNever married       -1.1963     0.5482  -2.182   0.0481 *
#> maritalPreviously married  -0.5038     0.4258  -1.183   0.2579
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 175.0038)
#>
#> Number of Fisher Scoring iterations: 2
```

Multivariate Regression

A final, more complex model is fitted, which includes multiple demographic variables and also considers the survey design.

```r
analytic.design2 <- svydesign(strata=~SDMVSTRA, id=~SDMVPSU, weights=~WTMEC2YR, data=analytic.d
summary(fit1 <- svyglm(blood.pressure ~ race + age.centred + gender + marital, design=analytic
#>
#> Call:
#> svyglm(formula = blood.pressure ~ race + age.centred + gender +
#>     marital, design = analytic.design2)
#>
#> Survey design:
#> svydesign(strata = ~SDMVSTRA, id = ~SDMVPSU, weights = ~WTMEC2YR,
#>     data = analytic.data, nest = TRUE)
#>
#> Coefficients:
#>                           Estimate Std. Error t value Pr(>|t|)
```

```
#> (Intercept)               71.14686    0.37976 187.348 3.26e-14 ***
#> raceNon-Hispanic Black     2.30248    0.84786   2.716 0.029953 *
#> raceNon-Hispanic White     1.08919    0.35921   3.032 0.019055 *
#> raceOther Hispanic         0.07339    1.20016   0.061 0.952947
#> raceOther race             2.11590    0.45579   4.642 0.002363 **
#> age.centred               -0.02223    0.01566  -1.419 0.198767
#> genderFemale              -2.91901    0.43979  -6.637 0.000294 ***
#> maritalNever married      -1.74002    0.48086  -3.619 0.008526 **
#> maritalPreviously married  0.22443    0.45139   0.497 0.634278
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 171.5455)
#>
#> Number of Fisher Scoring iterations: 2
```

A note about Predictive models

In statistical analyses involving survey data, it's crucial to account for the survey's design features. These features can include sampling weights, stratification, and clustering, among others. Ignoring these could lead to biased estimates and incorrect conclusions. Considering such survey design features make the analysis more robust and reliable in terms of inference.

However, when the goal shifts from inference to prediction, additional challenges come into play. Specifically, the model may perform well on the data used to fit it (the "training" data) but not generalize well to new, unseen data. This discrepancy between training performance and generalization to new data is often referred to as "overfitting," and the optimism of the model refers to the extent to which it overestimates its predictive performance on new data based on its performance on the training data.

Optimism-correction techniques are methodologies designed to address this issue. They allow you to evaluate how well your model is likely to perform on new data, not just the data you used to build it. Methods for optimism correction often involve techniques like cross-validation, bootstrapping, or specialized

types of model validation that help in estimating the 'true' predictive performance of the model. Some of these techniques were discussed in the predictive modelling chapter.

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

NHANES: Cholesterol

```r
# Load required packages
library(survey)
library(Publish)
library(tableone)
library(ROCR)
library(WeightedROC)
library(jtools)
library(dplyr)
```

Preprocessing

Analytic data set

We will use `cholesterolNHANES15part1.RData` in this prediction goal question (predicting cholesterol in adults).

For this exercise, we are assuming that:

- outcome: cholesterol
- predictors:
  - gender
  - whether born in US
  - race
  - education
  - whether married
  - income level
  - BMI
  - whether has diabetes
- survey features:

  - survey weights
  - strata
  - cluster/PSU; where strata is nested within clusters

- restrict to those participants who are of 18 years of age or older

```
load("Data/surveydata/cholesterolNHANES15part1.rdata") #Loading the dataset
ls()
#>  [1] "analytic"           "analytic.with.miss" "analytic1"
#>  [4] "analytic2"          "analytic2b"         "analytic3"
#>  [7] "collinearity"       "correlationMatrix"  "diff.boot"
#> [10] "extract.boot.fun"   "extract.fit"        "extract.lm.fun"
#> [13] "fictitious.data"    "fit0"               "fit1"
#> [16] "fit2"               "fit3"               "fit4"
#> [19] "fit5"               "formula0"           "formula1"
#> [22] "formula2"           "formula3"           "formula4"
#> [25] "formula5"           "has_annotations"    "k.folds"
#> [28] "numeric.names"      "perform"            "pred.y"
#> [31] "rocobj"             "sel.names"          "var.cluster"
#> [34] "var.summ"           "var.summ2"
```

Retaining only useful variables

```
# Data dimensions
dim(analytic)
#> [1] 1267   33

# Variable names
names(analytic)
#>  [1] "ID"                    "gender"                "age"
#>  [4] "born"                  "race"                  "education"
#>  [7] "married"               "income"                "weight"
#> [10] "psu"                   "strata"                "diastolicBP"
#> [13] "systolicBP"            "bodyweight"            "bodyheight"
#> [16] "bmi"                   "waist"                 "smoke"
#> [19] "alcohol"               "cholesterol"           "cholesterolM2"
#> [22] "triglycerides"         "uric.acid"             "protein"
#> [25] "bilirubin"             "phosphorus"            "sodium"
```

```
#> [28] "potassium"             "globulin"               "calcium"
#> [31] "physical.work"         "physical.recreational" "diabetes"

#Subsetting dataset with variables needed:
require(dplyr)
anadata <- select(analytic,
                  cholesterol, #outcome
                  gender, age, born, race, education, married, income, bmi, diabetes, #predicto
                  weight, psu, strata) #survey features

# new data sizes
dim(anadata)
#> [1] 1267   13

# retained variable names
names(anadata)
#>  [1] "cholesterol" "gender"      "age"         "born"        "race"
#>  [6] "education"   "married"     "income"      "bmi"         "diabetes"
#> [11] "weight"      "psu"         "strata"

#Restricting to participants who are 18 or older
summary(anadata$age) #The age range is already 20-80
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   20.00   36.00   51.00   49.91   63.00   80.00

#Recoding the born variable
table(anadata$born, useNA = "always")
#>
#> Born in 50 US states or Washingt                          Others
#>                              991                             276
#>                             <NA>
#>                                0
levels(anadata$born)
#> NULL
anadata$born <- car::recode(anadata$born,
                            "'Born in 50 US states or Washingt' = 'Born.in.US';
                            'Others' = 'Others';
                            else=NA")
table(anadata$born, useNA = "always")
#>
```

```
#> Born.in.US     Others       <NA>
#>        991        276          0
```

Checking the data for missing

```
require(DataExplorer)
plot_missing(anadata) #no missing data
```

Preparing factor and continuous variables appropriately

```
vars = c("cholesterol", "gender", "born", "race", "education",
         "married", "income", "bmi", "diabetes")
numeric.names <- c("cholesterol", "bmi")
factor.names <- vars[!vars %in% numeric.names]

anadata[factor.names] <- apply(X = anadata[factor.names],
                               MARGIN = 2, FUN = as.factor)

anadata[numeric.names] <- apply(X = anadata[numeric.names],
                                MARGIN = 2, FUN =function (x)
                                  as.numeric(as.character(x)))
```

Table 1

```r
library(tableone)
tab1 <- CreateTableOne(data = anadata, includeNA = TRUE, vars = vars)
print(tab1, showAllLevels = TRUE,  varLabels = TRUE)
#>
#>                            level              Overall
#>   n                                              1267
#>   cholesterol (mean (SD))                    193.10 (43.22)
#>   gender (%)               Female                496 (39.1)
#>                            Male                  771 (60.9)
#>   born (%)                 Born.in.US            991 (78.2)
#>                            Others                276 (21.8)
#>   race (%)                 Black                 246 (19.4)
#>                            Hispanic              337 (26.6)
#>                            Other                 132 (10.4)
#>                            White                 552 (43.6)
#>   education (%)            College               648 (51.1)
#>                            High.School           523 (41.3)
#>                            School                 96 ( 7.6)
#>   married (%)              Married               751 (59.3)
#>                            Never.married         226 (17.8)
#>                            Previously.married    290 (22.9)
#>   income (%)               <25k                  344 (27.2)
#>                            Between.25kto54k      435 (34.3)
#>                            Between.55kto99k      297 (23.4)
#>                            Over100k              191 (15.1)
#>   bmi (mean (SD))                               29.58 (6.84)
#>   diabetes (%)             No                   1064 (84.0)
#>                            Yes                   203 (16.0)
```

Linear regression when cholesterol is continuous

Fit a linear regression, and report the VIFs.

```r
#Fitting initial regression

fit0 <- lm(cholesterol ~ gender + born + race + education +
              married + income + bmi + diabetes,
```

```r
            data = anadata)

library(Publish)
publish(fit0)
#>     Variable              Units Coefficient           CI.95    p-value
#>  (Intercept)                         198.90 [184.82;212.97]    < 1e-04
#>       gender             Female         Ref
#>                            Male       -6.82  [-11.76;-1.89]   0.006854
#>         born         Born.in.US         Ref
#>                          Others       15.65    [8.54;22.75]    < 1e-04
#>         race              Black         Ref
#>                       Hispanic       -2.75   [-10.61;5.10]   0.492333
#>                          Other       -3.95   [-13.61;5.72]   0.423740
#>                          White        5.36   [-1.20;11.92]   0.109403
#>    education            College         Ref
#>                     High.School        3.51    [-1.61;8.63]   0.179871
#>                          School        0.31   [-9.63;10.24]   0.951841
#>      married            Married         Ref
#>                   Never.married      -11.05  [-17.67;-4.44]   0.001082
#>              Previously.married        4.72   [-1.43;10.86]   0.132468
#>       income               <25k         Ref
#>                Between.25kto54k       -0.48    [-6.72;5.75]   0.879480
#>                Between.55kto99k        3.41   [-3.60;10.43]   0.340491
#>                        Over100k        2.24   [-6.02;10.51]   0.595131
#>          bmi                          -0.21    [-0.56;0.15]   0.257105
#>     diabetes                 No         Ref
#>                             Yes      -10.61  [-17.21;-4.02]   0.001652

#Checking VIFs
car::vif(fit0)
#>               GVIF Df GVIF^(1/(2*Df))
#> gender    1.065810  1        1.032381
#> born      1.578258  1        1.256288
#> race      1.684064  3        1.090753
#> education 1.280113  2        1.063683
#> married   1.225520  2        1.052156
#> income    1.277005  3        1.041595
#> bmi       1.086953  1        1.042570
#> diabetes  1.073619  1        1.036156
```

All VIFs are small.

Test of association when cholesterol is binary

Dichotomize the outcome such that cholesterol<200 is labeled as 'healthy'; otherwise label it as 'unhealthy', and name it 'cholesterol.bin'. Test the association between this binary variable and gender.

```
#Creating binary variable for cholesterol
anadata$cholesterol.bin <- ifelse(anadata$cholesterol <200, "healthy", "unhealthy")
#If cholesterol is <200, then "healthy", if not, "unhealthy"

table(anadata$cholesterol.bin)
#>
#>   healthy unhealthy
#>       738       529
anadata$cholesterol.bin <- as.factor(anadata$cholesterol.bin)
anadata$cholesterol.bin <- relevel(anadata$cholesterol.bin, ref = "unhealthy")
```

Test of association between cholesterol and gender (no survey features)

```
# Simple Chi-square testing
chisq.chol.gen <- chisq.test(anadata$cholesterol.bin, anadata$gender)
chisq.chol.gen
#>
#>  Pearson's Chi-squared test with Yates' continuity correction
#>
#> data:  anadata$cholesterol.bin and anadata$gender
#> X-squared = 5.1321, df = 1, p-value = 0.02349
```

Setting up survey design

```r
require(survey)
summary(anadata$weight)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    5470   19540   30335   48904   63822  224892
w.design <- svydesign(id = ~psu, weights = ~weight, strata = ~strata,
                      nest = TRUE, data = anadata)
summary(weights(w.design))
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    5470   19540   30335   48904   63822  224892
```

Test of association accounting for survey design

```r
#Rao-Scott modifications (chi-sq)
svychisq(~cholesterol.bin + gender, design = w.design, statistic = "Chisq")
#>
#>  Pearson's X^2: Rao & Scott adjustment
#>
#> data:  svychisq(~cholesterol.bin + gender, design = w.design, statistic = "Chisq")
#> X-squared = 11.092, df = 1, p-value = 0.02365

#Thomas-Rao modifications (F)
svychisq(~cholesterol.bin + gender, design = w.design, statistic = "F")
#>
#>  Pearson's X^2: Rao & Scott adjustment
#>
#> data:  svychisq(~cholesterol.bin + gender, design = w.design, statistic = "F")
#> F = 5.1205, ndf = 1, ddf = 15, p-value = 0.03891
```

All three tests indicate strong evidence to reject the H0. There seems to be an association between gender and cholesterol level (healthy/unhealthy)

Table 1

Create a Table 1 (summarizing the covariates) stratified by the binary outcome: cholesterol.bin, utilizing the above survey features.

```r
# Creating Table 1 stratified by binary outcome (cholesterol)
# Using the survey features

vars2 = c("gender", "born", "race", "education",
         "married", "income", "bmi", "diabetes")


kableone <- function(x, ...) {
  capture.output(x <- print(x, showAllLevels= TRUE, padColnames = TRUE, insertLevel = TRUE))
  knitr::kable(x, ...)
}
kableone(svyCreateTableOne(var = vars2, strata= "cholesterol.bin", data=w.design, test = TRUE))
```

	level	unhealthy	healthy	p	test
n		27369732.3	34591444.0		
gender (%)	Female	13573865.5 (49.6)	13917447.5 (40.2)	0.039	
	Male	13795866.8 (50.4)	20673996.5 (59.8)		
born (%)	Born.in.US	23772751.7 (86.9)	31532673.3 (91.2)	0.028	
	Others	3596980.6 (13.1)	3058770.7 (8.8)		
race (%)	Black	1832118.3 (6.7)	3696893.4 (10.7)	0.015	
	Hispanic	3263992.3 (11.9)	3921344.6 (11.3)		
	Other	1887156.6 (6.9)	2601870.3 (7.5)		
	White	20386465.2 (74.5)	24371335.7 (70.5)		
education (%)	College	15855712.5 (57.9)	20945710.7 (60.6)	0.522	
	High.School	10615218.7 (38.8)	12434827.2 (35.9)		
	School	898801.1 (3.3)	1210906.1 (3.5)		
married (%)	Married	17489306.2 (63.9)	21170020.0 (61.2)	0.005	

	level	unhealthy	healthy	p	test
	Never.married	3086474.4 (11.3)	7175237.2 (20.7)		
	Previously.married	6703951.8 (24.8)	6246186.8 (18.1)		
income (%)	<25k	4760281.8 (17.4)	6364208.6 (18.4)	0.915	
	Between.25ktoless	8682481.6 (31.7)	10786198.6 (31.2)		
	Between.55ktoless	6919847.0 (25.4)	9190388.2 (26.6)		
	Over100k	6987121.9 (25.5)	8250648.6 (23.9)		
bmi (mean (SD))		29.35 (6.13)	29.64 (7.05)	0.593	
diabetes (%)	No	25080412.0 (91.6)	30006523.6 (86.7)	0.012	
	Yes	2289320.3 (8.4)	4584920.4 (13.3)		

Logistic regression model

Run a logistic regression model using the same variables, utilizing the survey features. Report the corresponding odds ratios and the 95% confidence intervals.

```
formula1 <- as.formula(I(cholesterol.bin=="unhealthy") ~ gender + born +
                         race + education + married + income + bmi +
                         diabetes)

fit1 <- svyglm(formula1,
               design = w.design,
               family = binomial(link = "logit"))

publish(fit1)
#>   Variable              Units OddsRatio       CI.95  p-value
#>     gender             Female       Ref
#>                          Male      0.70 [0.49;0.98]   0.2866
```

```
#>       born          Born.in.US       Ref
#>                        Others      2.10 [1.41;3.13]   0.1707
#>       race               Black       Ref
#>                     Hispanic      1.15 [0.80;1.67]   0.5871
#>                        Other      1.11 [0.69;1.80]   0.7406
#>                        White      1.46 [1.00;2.14]   0.3003
#>  education            College       Ref
#>                  High.School      1.21 [0.96;1.52]   0.3563
#>                       School      0.86 [0.52;1.43]   0.6712
#>    married            Married       Ref
#>                Never.married      0.54 [0.32;0.90]   0.2526
#>            Previously.married      1.31 [0.92;1.87]   0.3704
#>     income               <25k       Ref
#>              Between.25kto54k      1.03 [0.61;1.73]   0.9408
#>              Between.55kto99k      1.02 [0.66;1.56]   0.9525
#>                      Over100k      1.12 [0.73;1.72]   0.6920
#>        bmi                         1.00 [0.97;1.03]   0.9361
#>   diabetes                 No       Ref
#>                           Yes      0.62 [0.41;0.95]   0.2720
```

Wald test (survey version)

Perform a Wald test (survey version) to test the null hypothesis that all coefficients associated with the income variable are zero, and interpret.

```
#Testing the H0 that all coefficients associated with the income variable are zero
regTermTest(fit1, ~income)
#> Wald test for income
#>  in svyglm(formula = formula1, design = w.design, family = binomial(link = "logit"))
#> F =  0.1050099  on  3  and  1  df: p= 0.94611
```

The Wald test here gives a large p-value; We do not have evidence to reject the H0 of coefficient being 0. If the coefficient for income variable is 0, this means that the outcome in the model (cholesterol) is not affected by income. This suggests that removing income from the model does not statistically improve the model fit. So we can remove income variable from the model.

Backward elimination

Run a backward elimination (using the AIC criteria) on the above logistic regression fit (keeping important variables gender, race, bmi, diabetes in the model), and report the odds ratios and the 95% confidence intervals from the resulting final logistic regression fit.

```r
#Running backward elimination based on AIC
require(MASS)
scope <- list(upper = ~ gender + born + race + education +
                married + income + bmi + diabetes,
              lower = ~ gender + race + bmi + diabetes)

fit3 <- step(fit1, scope = scope, trace = FALSE,
                k = 2, direction = "backward")

#Odds Ratios
publish(fit3)
#>  Variable              Units OddsRatio       CI.95   p-value
#>    gender             Female       Ref
#>                         Male      0.71 [0.51;0.98]   0.08558
#>      born         Born.in.US       Ref
#>                       Others      2.01 [1.37;2.96]   0.01184
#>      race              Black       Ref
#>                     Hispanic      1.15 [0.81;1.65]   0.46785
#>                        Other      1.11 [0.68;1.81]   0.69539
#>                        White      1.46 [0.99;2.17]   0.10469
#>   married            Married       Ref
#>                Never.married      0.54 [0.32;0.90]   0.05770
#>           Previously.married      1.30 [0.93;1.80]   0.17125
#>       bmi                         1.00 [0.97;1.03]   0.95146
#>  diabetes                 No       Ref
#>                          Yes      0.61 [0.40;0.91]   0.05445
```

Born and married are also found to be useful on top of gender + race + bmi + diabetes.

Interaction terms

Checking interaction terms

– gender and whether married

– gender and whether born in the US

– gender and diabetes

– whether married and diabetes

```
#gender and married
fit4 <- update(fit3, .~. + interaction(gender, married))
anova(fit3, fit4)
#> Working (Rao-Scott+F) LRT for interaction(gender, married)
#>  in svyglm(formula = I(cholesterol.bin == "unhealthy") ~ gender +
#>     born + race + married + bmi + diabetes + interaction(gender,
#>     married), design = w.design, family = binomial(link = "logit"))
#> Working 2logLR =  0.7461308 p= 0.70903
#> (scale factors:  1.1 0.93 );  denominator df= 4
```

Do not include interaction term

```
#gender and born in us
fit5 <- update(fit3, .~. + interaction(gender, born))
anova(fit3, fit5)
#> Working (Rao-Scott+F) LRT for interaction(gender, born)
#>  in svyglm(formula = I(cholesterol.bin == "unhealthy") ~ gender +
#>     born + race + married + bmi + diabetes + interaction(gender,
#>     born), design = w.design, family = binomial(link = "logit"))
#> Working 2logLR =  0.4635299 p= 0.52441
#> df=1;  denominator df= 5
```

Do not include interaction term

```
#gender and diabetes
fit6 <- update(fit3, .~. + interaction(gender, diabetes))
anova(fit3, fit6)
#> Working (Rao-Scott+F) LRT for interaction(gender, diabetes)
#>  in svyglm(formula = I(cholesterol.bin == "unhealthy") ~ gender +
#>     born + race + married + bmi + diabetes + interaction(gender,
```

```
#>     diabetes), design = w.design, family = binomial(link = "logit"))
#> Working 2logLR =  1.222596 p= 0.32211
#> df=1;  denominator df= 5
```

Do not include interaction term

```
#married and diabetes
fit7 <- update(fit3, .~. + interaction(married, diabetes))
anova(fit3, fit7)
#> Working (Rao-Scott+F) LRT for interaction(married, diabetes)
#>  in svyglm(formula = I(cholesterol.bin == "unhealthy") ~ gender +
#>     born + race + married + bmi + diabetes + interaction(married,
#>     diabetes), design = w.design, family = binomial(link = "logit"))
#> Working 2logLR =  0.3207507 p= 0.84547
#> (scale factors:  1.4 0.62 );  denominator df= 4
```

Do not include interaction term

None of the interaction terms are improving the model fit.

AUC

Report AUC of the final model (only using weight argument) and interpret.

AUC of the final model (only using weight argument) and interpret

```
require(ROCR)
# WeightedROC may not be on cran for all R versions
# devtools::install_github("tdhock/WeightedROC")

library(WeightedROC)
svyROCw <- function(fit = fit3, outcome = anadata$cholesterol.bin == "unhealthy", weight = anad
  if (is.null(weight)){ # require(ROCR)
    prob <- predict(fit, type = "response")
  pred <- prediction(as.vector(prob), outcome)
  perf <- performance(pred, "tpr", "fpr")
  auc <- performance(pred, measure = "auc")
  auc <- auc@y.values[[1]]
```

```r
  roc.data <- data.frame(fpr = unlist(perf@x.values), tpr = unlist(perf@y.values),
      model = "Logistic")
  with(data = roc.data,plot(fpr, tpr, type="l", xlim=c(0,1), ylim=c(0,1), lwd=1,
     xlab="1 - specificity", ylab="Sensitivity",
     main = paste("AUC = ", round(auc,3))))
  mtext("Unweighted ROC")
  abline(0,1, lty=2)
  } else {
    outcome <- as.numeric(outcome)
  pred <- predict(fit, type = "response")
  tp.fp <- WeightedROC(pred, outcome, weight)
  auc <- WeightedAUC(tp.fp)
  with(data = tp.fp,plot(FPR, TPR, type="l", xlim=c(0,1), ylim=c(0,1), lwd=1,
     xlab="1 - specificity", ylab="Sensitivity",
     main = paste("AUC = ", round(auc,3))))
  abline(0,1, lty=2)
  mtext("Weighted ROC")
  }
}
svyROCw(fit = fit3, outcome = anadata$cholesterol.bin == "unhealthy", weight = anadata$weight)
```

The area under the curve in the final model is 0.611, using the survey weighted ROC. The AUC of 0.611 indicates that this model has poor discrimination.

Archer-Lemeshow Goodness of fit

Report Archer-Lemeshow Goodness of fit test and interpret (utilizing all the survey features).

```r
#Archer-Lemeshow Goodness of fit test utilizing all survey features
AL.gof2 <- function(fit = fit3, data = anadata,
                    weight = "weight", psu = "psu", strata = "strata"){
  r <- residuals(fit, type = "response")
  f<-fitted(fit)
  breaks.g <- c(-Inf, quantile(f, (1:9)/10), Inf)
  breaks.g <- breaks.g + seq_along(breaks.g) * .Machine$double.eps
  g<- cut(f, breaks.g)
  data2g <- cbind(data,r,g)
  newdesign <- svydesign(id=as.formula(paste0("~",psu)),
                         strata=as.formula(paste0("~",strata)),
                         weights=as.formula(paste0("~",weight)),
                         data=data2g, nest = TRUE)
  decilemodel <- svyglm(r~g, design=newdesign)
  res <- regTermTest(decilemodel, ~g)
  return(res)
}

AL.gof2(fit3, anadata, weight = "weight", psu = "psu", strata = "strata")
#> Wald test for g
#>  in svyglm(formula = r ~ g, design = newdesign)
#> F =  0.7569326  on  9  and  6  df: p= 0.66075
```

Archer and Lemeshow GoF test was used to test the fit of this model. The p-value of 0.3043, which is greater than 0.05. This means that there is no evidence of lack of fit to this model.

Add age as a predictor for linear regression

Fit another logistic regression (similar to Q1) with the above-mentioned predictors (as obtained in Q7) and age, utilizing the survey features. What difference do you see from the previous fit results?

```r
aic.int.model <- eval(fit3$call[[2]])
aic.int.model
#> I(cholesterol.bin == "unhealthy") ~ gender + born + race + married +
#>     bmi + diabetes

formula3 <- as.formula(cholesterol.bin ~ gender + born + race + married + bmi + diabetes + age)
fit9 <- svyglm(formula3,
               design = w.design,
               family = binomial(link="logit"))
summary(fit9)
#>
#> Call:
#> svyglm(formula = formula3, design = w.design, family = binomial(link = "logit"))
#>
#> Survey design:
#> svydesign(id = ~psu, weights = ~weight, strata = ~strata, nest = TRUE,
#>     data = anadata)
#>
#> Coefficients:
#>                              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)                 1.0657919  0.6260889   1.702   0.1494
#> genderMale                  0.3821902  0.1703394   2.244   0.0749 .
#> bornOthers                 -0.6912102  0.2026407  -3.411   0.0190 *
#> raceHispanic               -0.2442019  0.1823190  -1.339   0.2381
#> raceOther                  -0.1570271  0.2306208  -0.681   0.5262
#> raceWhite                  -0.3638735  0.2029676  -1.793   0.1330
#> marriedNever.married        0.4029107  0.2637962   1.527   0.1872
#> marriedPreviously.married -0.2096009  0.1620478  -1.293   0.2524
#> bmi                        -0.0002237  0.0134117  -0.017   0.9873
#> diabetesYes                 0.6534019  0.2456333   2.660   0.0449 *
#> age                        -0.0151364  0.0042038  -3.601   0.0155 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1.000456)
#>
#> Number of Fisher Scoring iterations: 4
publish(fit9)
#>  Variable              Units OddsRatio       CI.95   p-value
#>    gender             Female       Ref
```

```
#>                          Male      1.47 [1.05;2.05]   0.07487
#>      born          Born.in.US       Ref
#>                        Others      0.50 [0.34;0.75]   0.01902
#>      race               Black       Ref
#>                      Hispanic      0.78 [0.55;1.12]   0.23809
#>                         Other      0.85 [0.54;1.34]   0.52619
#>                         White      0.69 [0.47;1.03]   0.13299
#>   married             Married       Ref
#>                Never.married      1.50 [0.89;2.51]   0.18721
#>            Previously.married      0.81 [0.59;1.11]   0.25238
#>       bmi                         1.00 [0.97;1.03]   0.98734
#>  diabetes                  No       Ref
#>                           Yes      1.92 [1.19;3.11]   0.04488
#>       age                         0.98 [0.98;0.99]   0.01553
```

Comparing with previous model fit

```
publish(fit3)
#>  Variable              Units OddsRatio       CI.95   p-value
#>    gender             Female       Ref
#>                         Male      0.71 [0.51;0.98]   0.08558
#>      born          Born.in.US       Ref
#>                        Others      2.01 [1.37;2.96]   0.01184
#>      race               Black       Ref
#>                      Hispanic      1.15 [0.81;1.65]   0.46785
#>                         Other      1.11 [0.68;1.81]   0.69539
#>                         White      1.46 [0.99;2.17]   0.10469
#>   married             Married       Ref
#>                Never.married      0.54 [0.32;0.90]   0.05770
#>            Previously.married      1.30 [0.93;1.80]   0.17125
#>       bmi                         1.00 [0.97;1.03]   0.95146
#>  diabetes                  No       Ref
#>                           Yes      0.61 [0.40;0.91]   0.05445
publish(fit9)
#>  Variable              Units OddsRatio       CI.95   p-value
#>    gender             Female       Ref
#>                         Male      1.47 [1.05;2.05]   0.07487
#>      born          Born.in.US       Ref
```

```
#>                        Others      0.50 [0.34;0.75]   0.01902
#>      race               Black       Ref
#>                      Hispanic      0.78 [0.55;1.12]   0.23809
#>                         Other      0.85 [0.54;1.34]   0.52619
#>                         White      0.69 [0.47;1.03]   0.13299
#>   married             Married       Ref
#>                 Never.married      1.50 [0.89;2.51]   0.18721
#>            Previously.married      0.81 [0.59;1.11]   0.25238
#>       bmi                          1.00 [0.97;1.03]   0.98734
#>  diabetes                  No       Ref
#>                           Yes      1.92 [1.19;3.11]   0.04488
#>       age                          0.98 [0.98;0.99]   0.01553
AIC(fit3)
#>       eff.p         AIC    deltabar
#>   11.121795 1706.792543    1.235755
AIC(fit9)
#>      eff.p         AIC   deltabar
#>   12.14310 1694.15974    1.21431
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

NHANES: Subsetting

The tutorial demonstrates how to work with subset of complex survey data, specifically focusing on an NHANES example.

The required packages are loaded.

```
# Load required packages
library(survey)
library(Publish)
library(DataExplorer)
```

Load data

Survey data is loaded into the R environment.

```
load("Data/surveydata/NHANES17.RData")
ls()
#> [1] "analytic"           "analytic.with.miss" "has_annotations"
```

Check missingness

A subset of variables is selected, and the presence of missing data is visualized.

```
Vars <- c("ID",
          "weight",
          "psu",
          "strata",
          "gender",
          "born",
          "race",
          "bmi",
```

```
          "cholesterol",
          "diabetes")
analytic.full.data <- analytic.with.miss[,Vars]
```

A new variable is also created to categorize cholesterol levels as "healthy" or "unhealthy."

```
analytic.full.data$cholesterol.bin <- ifelse(analytic.full.data$cholesterol <200, "healthy", "u
analytic.full.data$cholesterol <- NULL

require(DataExplorer)
plot_missing(analytic.full.data)
```

Subsetting Complex Survey data

We are subsetting based on whether the subjects have missing observation (e.g., only retaining those with complete information). This is often an eligibility criteria in studies. In missing data analysis, we will learn more about more appropriate approaches.

```r
dim(analytic.full.data)
#> [1] 9254   10
head(analytic.full.data$ID) # full data
#> [1] 93703 93704 93705 93706 93707 93708
analytic.complete.case.only <- as.data.frame(na.omit(analytic.full.data))
dim(analytic.complete.case.only)
#> [1] 6636   10
head(analytic.complete.case.only$ID) # complete case
#> [1] 93705 93706 93707 93708 93709 93711
head(analytic.full.data$ID[analytic.full.data$ID %in% analytic.complete.case.only$ID])
#> [1] 93705 93706 93707 93708 93709 93711
```

Below we show how to identify who has missing observations vs not based on full (`analytic.full.data`) and complete case (`analytic.complete.case.only`) data. See Heeringa et al (2010) book page 114 (section 4.5.3 "Preparation for Subclass analyses") and also page 218 (section 7.5.4 "appropriate analysis: incorporating all Sample Design Features"). This is done for 2 reasons:

- full complex survey design structure is taken into account, so that variance estimation is done correctly. If one or more PSU were excluded because none of the complete cases were observed in those PSU, the sub-population (complete cases) will not have complete information of how many PSU were actually present in the original complex design. Then in the population, a reduced number of PSUs would be used to calculate variance (number of SPU is a component of the variance calculation formula, see equation (5.2) in Heeringa et al (2010) textbook. Same is true for strata.), and will result in a wrong/biased variance estimate. Also see West et al. doi: 10.1177/1536867X0800800404
- size of sub-population (here, those with complete cases) is recognized as a random variable; not just a fixed size.

```r
# assign missing indicator
analytic.full.data$miss <- 1
# assign missing indicator = 0 if the observation is available
analytic.full.data$miss[analytic.full.data$ID %in% analytic.complete.case.only$ID] <- 0
```

```r
table(analytic.full.data$miss)
#>
#>    0    1
#> 6636 2618
# IDs not in complete case data
head(analytic.full.data$ID[analytic.full.data$miss==1])
#> [1] 93703 93704 93710 93720 93724 93725
# IDs in complete case data
head(analytic.full.data$ID[analytic.full.data$miss==0])
#> [1] 93705 93706 93707 93708 93709 93711
```

Logistic regression on sub-population

A logistic regression model is run on the subset of data that has no missing values. Here, it distinguishes between correct and incorrect approaches to account for the complex survey design.

```r
require(survey)
require(Publish)
model.formula <- as.formula("I(cholesterol.bin=='healthy')~
                            diabetes+gender+born+race+bmi")
```

Wrong approach

```r
w.design.wrong <- svydesign(ids=~psu,
                        weights=~weight,
                        strata=~strata,
                        data = analytic.complete.case.only, #wrong!!
                        nest = TRUE)
```

Correct approach

```r
w.design0 <- svydesign(ids=~psu,
                        weights=~weight,
```

```
                        strata=~strata,
                        data = analytic.full.data,
                        nest = TRUE)

# retain only those that have complete observation / no missing
w.design <- subset(w.design0, miss == 0)# this is the subset design
```

Full model

```
fit <- svyglm(model.formula, family = quasibinomial,
              design = w.design) # subset design
publish(fit)
#>  Variable                            Units Coefficient           CI.95     p-value
#>  diabetes                               No         Ref
#>                                        Yes        0.38     [0.20;0.57]   0.0049202
#>    gender                            Female         Ref
#>                                         Male        0.22     [0.03;0.40]   0.0568343
#>      born Born in 50 US states or Washingt         Ref
#>                                      Others       -0.66   [-0.84;-0.47]   0.0002304
#>                                     Refused      -12.26 [-13.65;-10.88]     < 1e-04
#>      race                              Black         Ref
#>                                    Hispanic        0.20    [-0.08;0.47]   0.2075536
#>                                       Other       -0.17    [-0.38;0.03]   0.1439474
#>                                       White       -0.37   [-0.66;-0.09]   0.0355030
#>       bmi                                        -0.04   [-0.05;-0.02]   0.0007697
```

Variable selection

Finally, we discuss variable selection methods. We employ backward elimination to determine which variables are significant predictors while retaining an important variable in the model. If unsure about usefulness of some (gender, born, race, bmi) variables in predicting the outcome, check via backward elimination while keeping important variable (diabetes, say, that has been established in the literature) in the model

```r
model.formula <- as.formula("I(cholesterol.bin=='healthy')~
                            diabetes+gender+born+race+bmi")

scope <- list(upper = ~ diabetes+gender+born+race+bmi, lower = ~ diabetes)

fit <- svyglm(model.formula, design=w.design, # subset design
              family=quasibinomial)

fitstep <- step(fit,  scope = scope, trace = FALSE, direction = "backward")
publish(fitstep) # final model
#>  Variable                           Units Coefficient           CI.95     p-value
#>  diabetes                              No         Ref
#>                                       Yes        0.38     [0.20;0.57]   0.0049202
#>    gender                           Female         Ref
#>                                       Male        0.22     [0.03;0.40]   0.0568343
#>      born Born in 50 US states or Washingt         Ref
#>                                     Others       -0.66   [-0.84;-0.47]   0.0002304
#>                                    Refused      -12.26 [-13.65;-10.88]     < 1e-04
#>      race                             Black         Ref
#>                                   Hispanic        0.20    [-0.08;0.47]   0.2075536
#>                                      Other       -0.17    [-0.38;0.03]   0.1439474
#>                                      White       -0.37   [-0.66;-0.09]   0.0355030
#>       bmi                                       -0.04   [-0.05;-0.02]   0.0007697
```

Also see (Stata 2023) for further details.

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

References

R functions (D)

The list of new R functions introduced in this *Survey data analysis* lab component are below:

Function_name	Package_name	Use
AIC	base/stats	To extract the AIC value of a model
as.character	base	To create a character vector
as.numeric	base	To create a numeric vector
eval	base	To evaluate an expression
fitted	base/stats	To extract fitted values of a model
ls	base	To see the list of objects
psrsq	survey	To compute the Nagelkerke and Cox-Snell pseudo R-squared statistics for
regTermTest	survey	To test for an additional variable in a regression model
residuals	base/stats	To extract residuals of a model
stepAIC	MASS	To choose a model by stepwise AIC
step	base/stats	To choose a model by stepwise AIC but it can keep the pre-specified varia
summ	jtools	To show/publish regression tables
svyboxplot	survey	To produce a box plot for survey data
svyby	survey	To see the summary statistics for a survey design
svychisq	survey	To test the bivariate assocaition between two categorical variables for sur
svyCreateTableOne	tableone	To create a frequency table with a survey design
svydesign	survey	To create a design for the survey data analysis
svyglm	survey	To run design-adjusted generalized linear models
update	base/stats	To update and re-fit a regression model

Quiz (D)

> ⚠ Warning
>
> We are currently in the beta testing phase for our quiz page updates, so please be aware that some responses may not be accurate. Thank you for your patience.
> For those interested in downloading the quiz questions, you'll find a section below dedicated to downloading.

Live Quiz

You can click on the live quiz options to either see hints of why your response may be incorrect (response marked in red), or whether your choice of response is correct (response marked in green):

Download Quiz

1. **Downloading the File:**
   - Navigate to the link: See here.
   - Right-click on the link and select "Save link as…" from the dropdown menu.
   - Alternatively click here for downloading the quizzes.
   - Choose a destination folder on your computer where you'd like to save the file (e.g., Desktop). **Remember this location**, as you'll need to navigate to it later.
2. **Setting Up RStudio:**

- If you don't have RStudio installed, see the download link in here.
- Launch RStudio after installing.

3. **Installing Necessary R Packages:**

   - Before running the R Markdown file, ensure you have the required packages. In RStudio's console (located at the bottom left by default), enter the following commands to install them:

```
install.packages("learnr")
install.packages("xfun")
```

4. **Opening and Running the File:**

   - In RStudio, go to `File` > `Open File...`.
   - Navigate to the folder where you saved the `Rmd` file and select it to open.
   - Once the file is open in RStudio, you'll see a "Run Document" button (green) at the top of the script editor. Click on it to run the R Markdown Quiz file.

App (D)

Below is an example of an app that utilizes the NHANES dataset following the tutorial materials. Users can choose variables for bivariate analysis, and decide whether to apply survey features or not.

> Tip
>
> **Web App may need some time to load the necessary files, depending on your internet speed.**

> Important
>
> If you want to run the app locally in your computer instead for faster load time: you can download the app directly in your R via `shiny::runGitHub("EpiShinyLiveD", "ehsanx", subdir = "app")`. Make sure that `shiny`, `survey` and `publish` packages are installed in your computer/R.

> Warning
>
> **Bug Report**:
> Fill out this form to report any issues with the app.

Exercise (D)

Tip

You can download all of the related files in a zip file **surveydataEx.zip** from Github folder, or just by clicking this link directly.

- Navigate to the GitHub folder (above link) where the ZIP file is located.
- Click on the file name (above zip file) to open its preview window.
- Click on the Download button to download the file. If you can't see the Download button, click on "Download Raw File" link that should appear on the page.

Problem Statement

We will revisit the article by Flegal et al. (2016). Our primary aim this time is to execute the survey data analysis more rigorously, specifically by incorporating essential survey features into our analysis.

We will reproduce some results from the article. The authors used NHANES 2013-14 dataset to create their main analytic dataset. The dataset contains 10,175 subjects with 12 relevant variables:

This is the same article that we discussed in our data access chapter!

- SEQN: Respondent sequence number
- RIDAGEYR: Age in years at screening
- RIAGENDR: Gender
- DMDEDUC2: Education level

- RIDRETH3: Race/ethnicity
- RIDEXPRG: Pregnancy status at exam
- WTINT2YR: Full sample 2 year weights
- SDMVPSU: Masked variance pseudo-PSU
- SDMVSTRA: Masked variance pseudo-stratum
- BMXBMI: Body mass index in kg/m**2
- SMQ020: Whether smoked at least 100 cigarettes in life
- SMQ040: Current status of smoking (Do you now smoke cigarettes?)

Question 1: Creating data and table

1(a) Importing dataset

```r
# you have to download the data in the same folder
load("Data/surveydata/Flegal2016.RData")
ls()
#> [1] "dat.full"        "has_annotations"
names(dat.full)
#>  [1] "SEQN"     "RIDAGEYR" "RIAGENDR" "DMDEDUC2" "RIDRETH3" "RIDEXPRG"
#>  [7] "WTINT2YR" "SDMVPSU"  "SDMVSTRA" "BMXBMI"   "SMQ020"   "SMQ040"
```

1(b) Subsetting according to eligibility

Subset the dataset according to the eligibility criteria described in the second paragraph of the `Methods` section.

- Hint: The authors restricted their study to
  - adults aged 20 years and more,
  - non-missing body mass index, and
  - non-pregnant.

Your analytic sample size should be 5,455, as described in the first sentence in the `Results` section.

```r
# 20+
dat.analytic <- subset(dat.full, RIDAGEYR>=20) # N = 5,769

# Non-missing outcome
dat.analytic <- subset(dat.analytic, !is.na(BMXBMI)) # N = 5,520

# Non-pregnant
dat.analytic <- subset(dat.analytic, is.na(RIDEXPRG) | RIDEXPRG !=
                         "Yes, positive lab pregnancy test") # N = 5,455

dim(dat.analytic)
#> [1] 5455   12
```

1(c) Reproduce Table 1

Reproduce Table 1 of the article.

- Hint 1: The authors reported unweighted frequencies, and thus, survey features should not be utilized to answer this question. Please be advised to order the categories as shown in the table. `tableone` package could be helpful.
- Hint 2: the authors did not show the results for the `Other` race category. But in your table, you could include all race categories.

```r
library(tableone)

dat <- dat.analytic

# Age
dat$age <- cut(dat$RIDAGEYR, c(20, 40, 60, Inf), right = FALSE)

# Gender
dat$gender <- dat$RIAGENDR

# Race/Hispanic origin group
dat$race <- dat$RIDRETH3
dat$race <- car::recode(dat$race, " 'Non-Hispanic White'='White'; 'Non-Hispanic Black'=
                        'Black'; 'Non-Hispanic Asian'='Asian'; c('Mexican American',
```

```r
                        'Other Hispanic')='Hispanic'; 'Other Race - Including Multi-Rac'=
                        'Other'; else=NA", levels = c("White", "Black", "Asian",
                                                       "Hispanic", "Other"))

# Table 1: Overall
tab11 <- CreateTableOne(vars = "age", strata = "race", data = dat, test = F,
                        addOverall = T)

# Table 1: Male
tab12 <- CreateTableOne(vars = "age", strata = "race", test = F, addOverall = T,
                        data = subset(dat, gender == "Male"))

# Table 1: Female
tab13 <- CreateTableOne(vars = "age", strata = "race", test = F, addOverall = T,
                        data = subset(dat, gender == "Female"))

# Reproducing Table 1
tab1a <- list(Overall = tab11, Male = tab12, Female = tab13)
print(tab1a, format = "f") # Showing only frequencies
#> $Overall
#>              Stratified by race
#>               Overall White Black Asian Hispanic Other
#>   n           5455    2343  1115  623   1214     160
#>   age
#>      [20,40)  1810     734   362  216    412      86
#>      [40,60)  1896     759   383  251    449      54
#>      [60,Inf) 1749     850   370  156    353      20
#>
#> $Male
#>              Stratified by race
#>               Overall White Black Asian Hispanic Other
#>   n           2638    1130  556   300   573      79
#>   age
#>      [20,40)   909     386  182   106   189      46
#>      [40,60)   897     360  179   120   215      23
#>      [60,Inf)  832     384  195    74   169      10
#>
#> $Female
#>              Stratified by race
#>               Overall White Black Asian Hispanic Other
```

```
#>   n           2817    1213  559   323   641      81
#>   age
#>      [20,40)   901     348  180   110   223      40
#>      [40,60)   999     399  204   131   234      31
#>      [60,Inf)  917     466  175    82   184      10
```

Question 2

2(a) Reproduce Table 1 with survey features [15% grade]

Not in this article but in many other articles, you would see `n` comes from the analytic sample and `%` comes from the survey design that accounts for survey features such as strata, clusters and survey weights. In Question 1, you see how n comes from the analytic sample. Your task for Question 2(a) is to create % part of the Table 1 with survey features, i.e., % should come from the survey design that accounts for strata, clusters and survey weights.

- Hint 1: Subset the design, not the sample. If you have generated a variable in your analytic dataset (based on eligibility), that variable should also be present in the full dataset.
- Hint 2: Generate age, gender, and race variable in your full data (codes shown in Question 1 could be helpful).
- Hint 3: Subset the design.
- Hint 4: Reproduce Table 1 with the design. `svyCreateTableOne` could be a helpful function.

```r
## Create all variables in the full data
# Age
dat.full$age <- cut(dat.full$RIDAGEYR, c(20, 40, 60, Inf), right = FALSE)

# Gender
dat.full$gender <- dat.full$RIAGENDR

```

```r
# Race/Hispanic origin group
dat.full$race <- dat.full$RIDRETH3
dat.full$race <- car::recode(dat.full$race, " 'Non-Hispanic White'='White';
                             'Non-Hispanic Black'='Black'; 'Non-Hispanic Asian'='Asian';
                             c('Mexican American','Other Hispanic')='Hispanic';
                             'Other Race - Including Multi-Rac'='Other';
                             else=NA", levels = c("White", "Black", "Asian",
                                                  "Hispanic", "Other"))

## Subset the design
# your codes here


## Table 1
# your codes here

#print(tab1b, format = "p") # Showing only percentages
```

2(b) Reproduce Table 3 [50% grade]

Reproduce the first column of Table 3 of the article (i.e., among men, explore the relationship between obesity and four predictors shown in the table).

- Hint 1: If necessary, re-level or re-order the levels. Use `Publish` package to report the estimates.
- Hint 2: Subset the design, not the sample. If you have generated a variable in your analytic dataset (based on eligibility), that variable should also be present in the full dataset.
- Hint 3: The authors used SAS to produce the results vs. We are using R. The estimates could be slightly different (in second decimal point) from the estimates presented in Table 3, but they should be approximately similar.
- Hint 4: You need to generate two variables, `smoking status` and `education`. The unweighted frequencies

should be matched with the frequencies in eTable 1 and eTable 2.

Your odds ratios could be look like as follows:

Variable	Units	OddsRatio
age	[20,40)	Ref
	[40,60)	1.28
	[60,Inf)	1.20
race	White	Ref
	Black	1.24
	Asian	0.27
	Hispanic	1.22
	Other	1.23
smoking	Never smoker	Ref
	Former smoker	1.25
	Current smoker	0.71
education	High school	Ref
	<High school	0.93
	>High school	0.97

```
## Recode Obese, Smoking status, Education - work on full data
# your codes here


## Set up the survey design
# your codes here


## Reproduce Table 3 - column 1
# your codes here
```

2(c) Model selection [25% grade]

From the literature, you know that `age` and `race` needs to be adjusted in the model, but you are not sure about `smoking` and `education`. Run an AIC based backward selection process to figure out whether you want to add `smoking` or `education`,

or both in the final model in 2(b). What is your conclusion [Expected answer: one short sentence]?

- Hint 1: You need to make sure your design (that is based on eligibility) is free from missing values. Even after applying eligibility criteria, you may have some missing values on multiple variables (see eTable 1 and eTable 2). This is especially important for model selection process.
- Hint 2: Work with the analytic data, keep only the relevant variables, and then remove missing values. Finally, subset the design and then select your final model.

```
## Recode Obese, Smoking status, Education - work on analytic data
# your codes here


## Remove missing values - work on analytic data
# your codes here


## Set up the survey design
# your codes here


## Model selection
# your codes here
```

2(d) Testing for interactions [10% grade]

Check whether the interaction between `age` and `smoking` should be added in the 2(b) model (yes or no answer required, along with the code and p-value):

```
# your codes here
```

Part VII

Missing data analysis

Background

The chapter provides a comprehensive guide on missing data analysis, emphasizing various imputation techniques to address data gaps. It begins by introducing the concept of imputation and the different types of missing data: MCAR, MAR, and MNAR. The tutorial then delves into multiple imputation methods for complex survey data, highlighting the importance of visualizing missing data patterns, creating multiple imputed datasets, and pooling results for a consolidated analysis. The challenges of imputing dependent and exposure variables are addressed, with a focus on the benefits of using auxiliary variables. The guide also explores the estimation of model performance in datasets with missing values, using metrics like the AUC and the Archer-Lemeshow test. Special attention is given to handling subpopulations with missing observations, testing the MCAR assumption empirically, and understanding effect modification within multiple imputation.

> **!** Important
>
> **Datasets**:
> All of the datasets used in this tutorial can be accessed from this GitHub repository folder

In the vast landscape of survey data analysis, one challenge consistently emerges as both a hurdle and an opportunity: missing data. As we delve into this chapter, we'll confront the often-encountered issue of incomplete or absent data points in survey datasets. Missing data isn't just a challenge; it's an invitation to refine our analytical techniques. This chapter will equip you with the tools and methodologies to handle missing data adeptly, ensuring that our survey data analysis remains robust and reliable.

Overview of tutorials

Missing data and imputation

Imputation is a technique used to replace missing data with substituted values. In health research, missing data is a common issue, and imputation helps in ensuring datasets are complete, leading to more accurate analyses. There are three types of missing data: Missing Completely at Random (MCAR), Missing at Random (MAR), and Missing Not at Random (MNAR). The type of missingness determines how the missing data should be handled. Various single imputation methods, such as mean imputation, regression imputation, and

predictive mean matching, are used based on the nature of the missing data. Multiple imputation is a process where the incomplete dataset is imputed multiple times, and the results are pooled together for more accurate analyses. Variable selection is crucial when analyzing missing data, and methods like majority, stack, and Wald are used to determine the best model. It's also essential to assess the impact of missing values on model fitting (convergence and diagnostics) to ensure the reliability of the results.

Multiple imputation in complex survey data

In the tutorial involve understanding how to assess the missing data patterns and visualize to understand the extent of missingness. Multiple imputations are then performed to address the missing data, creating multiple versions of the dataset with varying imputations. After imputation, new variables are created or modified for analysis, and the integrity of the imputed data is checked both visually. The tutorial also emphasizes the importance of combining multiple imputed datasets for analysis. Logistic regression is applied to the imputed datasets, and the results are pooled to get a single set of estimates. The tutorial concludes with a variable selection process to identify the most relevant variables for the model.

Multiple imputation then deletion (MID)

This tutorial emphasizes the challenges of imputing dependent and exposure variables. The tutorial underscores the potential benefits of using auxiliary variables in the imputation process. While traditional Multiple Imputation (MI) and MID can yield similar results, MID is particularly advantageous when there's a significant percentage of missing values in the outcome variable. The tutorial walks through the process of data loading, identifying missing values, performing standard imputations, and adding missing indicators. Subsequent steps involve structuring the data for survey design, fitting statistical models to each imputed dataset, and pooling the results for a consolidated

analysis. The final stages focus on calculating and presenting odds ratios to interpret the relationships between variables.

Model performance from multiple imputed datasets

In the context of survey data analysis, the provided tutorial outlines the process of estimating model performance, particularly when dealing with weighted data that has missing values. The focus is about estimating treatment effects, both individually and in a pooled manner. Model performance is gauged using the Area Under the Curve (AUC) and the Archer-Lemeshow (AL) test. This is done for models with and without interactions. The results provide insights into the model's accuracy and fit, with the AUC offering a measure of the model's discriminative ability and the AL test indicating the model's goodness of fit to the data. The appendix provides a closer look at the user-defined functions used throughout the analysis.

Dealing with subpopulations with missing observations

The primary objective is to showcase how to handle missing data analysis with multiple imputation in the backdrop of complex surveys, particularly when we are interested in subpopulations. The process involves working with the analytic data, imputing missing values from this dataset, accounting for ineligible subjects from the complete data, and reincorporating these ineligible subjects into the imputed datasets. This ensures that the survey's features can be utilized and the design subsetted accordingly. After importing and inspecting the dataset, the analysis subsets the data based on eligibility criteria, imputes missing values, and prepares the survey design. The subsequent steps involve design-adjusted logistic regression and pooling of estimates using Rubin's rule.

Testing MCAR assumption empirically in the data

The tutorial discusses the process of testing for Missing Completely At Random (MCAR) in datasets. Initially, essential packages are loaded to facilitate the analysis. A DAG is defined

to represent the causal relationships between dataset variables, and this DAG is used to simulate a dataset. The DAG is then visualized, and the simulated dataset undergoes random data omission to mimic MCAR scenarios. Various visualizations, such as margin plots, are employed to understand the distribution of missing values in relation to other variables. Little's MCAR test, a statistical method, is applied to determine if the data is indeed MCAR. The test's limitations are also discussed. Additionally, tests for multivariate normality and homoscedasticity are conducted. In a subsequent section, data is intentionally set to missing based on a specific rule, and similar analyses and visualizations are performed to understand the nature of this missingness.

Effect modification within multiple imputation

The tutorial delves into the intricacies of effect modification within the realm of survey data analysis. A dataset is comprising several imputed datasets is used. The primary objective is to investigate how two specific variables interact in predicting a particular outcome. To this end, logistic regression models are constructed for each level of a categorical variable. Emphasis is placed on the significance of Odds Ratios (ORs) in interpreting these interactions. Subsequently, simple slopes analyses are performed for each imputed dataset, shedding light on the relationship between the predictor and the outcome at distinct levels of a moderating variable. The outcomes from each imputed dataset are then pooled to offer a comprehensive understanding of the effect modification.

> Tip
>
> **Optional Content**:
> You'll find that some sections conclude with an optional video walkthrough that demonstrates the code. Keep in mind that the content might have been updated since these videos were recorded. Watching these videos is optional.

Warning

Bug Report:
Fill out this form to report any issues with the tutorial.

Concepts (M)

Missing Data Analysis

This section is about understanding, categorizing, and addressing missing data in clinical and epidemiological research. It highlights the prevalence of missing data in these fields, the common use of complete case analysis without considering the implications, and the types of missingness: Missing Completely at Random (MCAR), Missing at Random (MAR), and Not Missing at Random (NMAR), each requiring different approaches and considerations. The consequences of not properly addressing missing data are detailed as bias, incorrect standard errors/precision, and a substantial loss of power.

This section also delves into strategies for addressing missing data, focusing on ad-hoc approaches and imputation methods. Ad-hoc approaches, such as ignoring missing data or using a missing category indicator, are generally dismissed as statistically invalid. In contrast, imputation, particularly multiple imputation (MI), is presented as a more robust and statistically sound method. Multiple imputation involves creating multiple complete datasets by predicting missing values and pooling the results to address the uncertainty associated with missing data. The section further discusses the types of imputation, the necessity of including a sufficient number of predictive variables, and the use of subject-area knowledge in building imputation models, providing a nuanced understanding of the challenges and solutions associated with missing data in research.

Reporting Guideline section delves into the complexities of handling missing data in statistical analysis, primarily through MI methods, especially Multiple Imputation by Chained Equations (MICE). It lays out the assumptions necessary for these methods (MCAR, MAR, MNAR). The guide also details how MICE

works, using sequential regression imputation to create multiple imputed datasets, thereby allowing for more accurate and robust statistical inferences. Additionally, it provides comprehensive instructions on reporting MICE analysis, including detailing the missingness rates, the reasons for missing data, the assumptions made, and the specifics of the imputation and pooling methods used, ensuring transparency and reproducibility in research.

Reading list

Key reference: (Sterne and al. 2009)

Optional reading: (Van Buuren 2018a)

Further optional readings: (Lumley 2011; Granger, Sergeant, and Lunt 2019; Hughes et al. 2019)

Video Lessons

Missing Data Analysis

Reporting guidelines when missing data is present

Video Lesson Slides

Missing data

Reporting guideline

Links

Video Lessons

- Google Slides
- PDF Slides

Reporting guideline

- Google Slides
- PDF Slides

References

Imputation

What is imputation?

Imputation is the process of replacing missing data with substituted values. In health research, it's common to have missing data. This tutorial teaches you how to handle and replace these missing values using the mice package in R.

Why is imputation important?

Missing data can lead to biased or incorrect results. Imputation helps in making the dataset complete, which can lead to more accurate analyses.

Key reference

In this discussion, our primary guide and source of information is the work titled "Flexible Imputation of Missing Data" by Stef van Buuren, denoted here as (Van Buuren 2018b). This book is an invaluable resource for anyone looking to delve deeper into the intricacies of handling missing data, especially in the context of statistical analyses. Below we also cited the relevant section numbers.

First, you need to load the necessary libraries:

```r
# Load required packages
library(mice)
library(DataExplorer)
library(VIM)
library(mitools)
```

Type of missing data

- Ref: (Van Buuren 2018b), Section 1.2

In this section, we are going to introduce three types of missing data that we will encounter in data analysis.

1. **Missing Completely at Random (MCAR)**:

The reason data is missing is completely random and not related to any measured or unmeasured variables. It's often an unrealistic assumption.

2. **Missing at Random (MAR)**:

The missing data is related to variables that are observed.

3. **Missing Not at Random (MNAR)**:

The missing data is related to variables that are not observed.

Why does missingness type matter?

The type of missingness affects how you handle the missing data:

- If data is MCAR, you can still analyze the complete cases without introducing bias.
- If data is MAR, you can use imputation to replace the missing values.
- If data is MNAR, it's challenging to address, and estimates will likely be biased. We could do some sensitivity analyses to check the impact.

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Data imputation

Getting to know the data

Before imputing, you should understand the data. The tutorial uses the `analytic.with.miss` dataset from NHANES. Various plots and functions are used to inspect the missing data pattern and relationships between variables.

> **Important**
>
> 1. Take a look here for those who are interested in how the analytic data was created.
> 2. For the purposes of this lab, we are just going to treat the data as SRS, and not going to deal with intricacies of survey data analysis.

```r
require(VIM)
load("Data/missingdata/NHANES17.RData")
NHANES17s <- analytic.with.miss[1:30,c("age", "bmi", "cholesterol","diastolicBP")]
NHANES17s
#>    age  bmi cholesterol diastolicBP
#> 1   NA 17.5          NA          NA
#> 2   NA 15.7          NA          NA
#> 3   66 31.7         157          NA
#> 4   NA 21.5         148          74
#> 5   NA 18.1         189          38
#> 6   66 23.7         209          NA
#> 7   75 38.9         176          66
#> 8   NA   NA          NA          NA
```

```
#> 9   56 21.3         238          68
#> 10  NA 19.7         182          68
#> 11  67 23.5         184          70
#> 12  54 39.9         230          NA
#> 13  71 22.5         180          60
#> 14  61 30.7         225          72
#> 15  22 24.5         213          62
#> 16  45 22.0         152          88
#> 17  NA 26.0          97          62
#> 18  NA   NA          NA          NA
#> 19  60 35.9         122          68
#> 20  60 23.8         184          68
#> 21  64 22.4         202          72
#> 22  NA 14.7          NA          NA
#> 23  NA 16.1          NA          NA
#> 24  67 31.1         176          52
#> 25  70 23.9         167          NA
#> 26  53 33.4         143          74
#> 27  42 27.6         165          86
#> 28  57 28.6         221          74
#> 29  20 27.6         153          54
#> 30  72 21.3          NA          76
NHANES17s[complete.cases(NHANES17s),]
#>    age  bmi cholesterol diastolicBP
#> 7   75 38.9         176          66
#> 9   56 21.3         238          68
#> 11  67 23.5         184          70
#> 13  71 22.5         180          60
#> 14  61 30.7         225          72
#> 15  22 24.5         213          62
#> 16  45 22.0         152          88
#> 19  60 35.9         122          68
#> 20  60 23.8         184          68
#> 21  64 22.4         202          72
#> 24  67 31.1         176          52
#> 26  53 33.4         143          74
#> 27  42 27.6         165          86
#> 28  57 28.6         221          74
#> 29  20 27.6         153          54
md.pattern(NHANES17s)
```

```
#>    bmi cholesterol age diastolicBP
#> 15   1           1   1           1  0
#> 4    1           1   1           0  1
#> 4    1           1   0           1  1
#> 1    1           0   1           1  1
#> 4    1           0   0           0  3
#> 2    0           0   0           0  4
#>      2           7  10          10 29
# Inspect the missing data pattern (each row = pattern)
# possible missingness (0,1) pattern and counts
# last col = missing counts for each variables
# last row = how many variable values missing in the row
# First col: Frequency of the pattern
# e,g, 2 cases missing for bmi

require(DataExplorer)
plot_missing(NHANES17s)
```

```r
# check the missingness

require(VIM)
marginplot(NHANES17s[, c("diastolicBP", "bmi")])
```

```r
marginplot(NHANES17s[, c("diastolicBP", "cholesterol")])
```

```r
marginplot(NHANES17s[, c("cholesterol", "bmi")])
```

```r
# distribution of observed data given the other variable is observed
# for MCAR, blue and red box plots should be similar
```

Single imputation

- Ref: (Van Buuren 2018b), Section 1.3

Impute NA only once. Below are some examples (Van Buuren and Groothuis-Oudshoorn 2011):

0.0.0.0.1 * Mean imputation

- Ref: (Van Buuren 2018b), Section 1.3.3, and (Buuren and Groothuis-Oudshoorn 2010)

Mean imputation is a straightforward method where missing values in a dataset are replaced with the mean of the observed values. While it's simple and intuitive, this approach can reduce the overall variability of the data, leading to an underestimation of variance. This can be problematic in statistical analyses where understanding data spread is crucial.

```
# Replace missing values by mean
imputation1 <- mice(NHANES17s,
                    method = "mean", # Replace by mean of the other values
                    m = 1, # Number of multiple imputations.
                    maxit = 1) # Number of iteration; mostly useful for convergence
#>
#>  iter imp variable
#>   1   1  age  bmi  cholesterol  diastolicBP
imputation1$imp
#> $age
#>       1
#> 1  57.4
#> 2  57.4
#> 4  57.4
#> 5  57.4
#> 8  57.4
#> 10 57.4
#> 17 57.4
#> 18 57.4
#> 22 57.4
#> 23 57.4
#>
```

```
#> $bmi
#>           1
#> 8  25.12857
#> 18 25.12857
#>
#> $cholesterol
#>           1
#> 1  178.8261
#> 2  178.8261
#> 8  178.8261
#> 18 178.8261
#> 22 178.8261
#> 23 178.8261
#> 30 178.8261
#>
#> $diastolicBP
#>       1
#> 1  67.6
#> 2  67.6
#> 3  67.6
#> 6  67.6
#> 8  67.6
#> 12 67.6
#> 18 67.6
#> 22 67.6
#> 23 67.6
#> 25 67.6
complete(imputation1, action = 1) # this is a function from mice
#>     age      bmi cholesterol diastolicBP
#> 1  57.4 17.50000    178.8261        67.6
#> 2  57.4 15.70000    178.8261        67.6
#> 3  66.0 31.70000    157.0000        67.6
#> 4  57.4 21.50000    148.0000        74.0
#> 5  57.4 18.10000    189.0000        38.0
#> 6  66.0 23.70000    209.0000        67.6
#> 7  75.0 38.90000    176.0000        66.0
#> 8  57.4 25.12857    178.8261        67.6
#> 9  56.0 21.30000    238.0000        68.0
#> 10 57.4 19.70000    182.0000        68.0
#> 11 67.0 23.50000    184.0000        70.0
```

```
#> 12 54.0 39.90000    230.0000        67.6
#> 13 71.0 22.50000    180.0000        60.0
#> 14 61.0 30.70000    225.0000        72.0
#> 15 22.0 24.50000    213.0000        62.0
#> 16 45.0 22.00000    152.0000        88.0
#> 17 57.4 26.00000     97.0000        62.0
#> 18 57.4 25.12857    178.8261        67.6
#> 19 60.0 35.90000    122.0000        68.0
#> 20 60.0 23.80000    184.0000        68.0
#> 21 64.0 22.40000    202.0000        72.0
#> 22 57.4 14.70000    178.8261        67.6
#> 23 57.4 16.10000    178.8261        67.6
#> 24 67.0 31.10000    176.0000        52.0
#> 25 70.0 23.90000    167.0000        67.6
#> 26 53.0 33.40000    143.0000        74.0
#> 27 42.0 27.60000    165.0000        86.0
#> 28 57.0 28.60000    221.0000        74.0
#> 29 20.0 27.60000    153.0000        54.0
#> 30 72.0 21.30000    178.8261        76.0
# there is another function in tidyr with the same name!
# use mice::complete() to avoid conflict
## the imputed dataset
```

0.0.0.0.2 * Regression Imputation

- Ref: (Van Buuren 2018b), Section 1.3.4

Regression imputation offers a more nuanced approach, especially when dealing with interrelated variables. By building a regression model using observed data, missing values are predicted based on the relationships between variables. This method can provide more accurate estimates for missing values by leveraging the inherent correlations within the data, making it a preferred choice in many scenarios over mean imputation.

$Y \sim X$

$age \sim bmi + cholesterol + diastolicBP$

```r
imputation2 <- mice(NHANES17s,
            method = "norm.predict", # regression imputation
            seed = 1,
            m = 1,
            print = FALSE)

# look at all imputed values
imputation2$imp
#> $age
#>           1
#> 1  55.32215
#> 2  54.99604
#> 4  55.65437
#> 5  53.68539
#> 8  56.70424
#> 10 55.79329
#> 17 54.38372
#> 18 56.70422
#> 22 54.81486
#> 23 55.06851
#>
#> $bmi
#>           1
#> 8  25.12857
#> 18 25.12857
#>
#> $cholesterol
#>           1
#> 1  183.3772
#> 2  184.2347
#> 8  179.7431
#> 18 179.7431
#> 22 184.7111
#> 23 184.0442
#> 30 183.4399
#>
#> $diastolicBP
#>           1
#> 1  66.48453
#> 2  66.24254
```

```
#> 3  68.82463
#> 6  67.47980
#> 8  67.51011
#> 12 68.91318
#> 18 67.51011
#> 22 66.10810
#> 23 66.29631
#> 25 67.93401

# examine the correlation between age and bmi before and after imputation
fit1 <- lm(age ~ bmi, NHANES17s)

summary(fit1) ## original data
#>
#> Call:
#> lm(formula = age ~ bmi, data = NHANES17s)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -37.383  -5.194   3.168   9.444  15.965
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  53.3482    17.1585   3.109  0.00606 **
#> bmi           0.1462     0.6063   0.241  0.81219
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 15.51 on 18 degrees of freedom
#>   (10 observations deleted due to missingness)
#> Multiple R-squared:  0.003219,   Adjusted R-squared:  -0.05216
#> F-statistic: 0.05814 on 1 and 18 DF,  p-value: 0.8122
sqrt(summary(fit1)$r.squared)
#> [1] 0.05674047

fit2 <- lm(age ~ bmi, mice::complete(imputation2))
summary(fit2) ## imputed complete data
#>
#> Call:
#> lm(formula = age ~ bmi, data = mice::complete(imputation2))
```

```
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -37.152  -1.407   0.000   8.026  15.989
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  52.1516     9.2485   5.639 4.86e-06 ***
#> bmi           0.1812     0.3568   0.508    0.616
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 12.45 on 28 degrees of freedom
#> Multiple R-squared:  0.009127,   Adjusted R-squared:  -0.02626
#> F-statistic: 0.2579 on 1 and 28 DF,  p-value: 0.6155
sqrt(summary(fit2)$r.squared)
#> [1] 0.09553283
## Relationship become stronger before imputation.
# with(data=NHANES17s, cor(age, bmi, use = "complete.obs"))

with(data=NHANES17s, cor(age, bmi, use = "pairwise.complete.obs"))
#> [1] 0.05674047
with(data = mice::complete(imputation2), cor(age, bmi))
#> [1] 0.09553283
```

0.0.0.0.3 * Stochastic regression imputation

- Ref: (Van Buuren 2018b), Section 1.3.5

Regression imputation, while powerful, has an inherent limitation. When it employs the fitted model to predict missing values, it does so without incorporating the error terms. This means that the imputed values are precisely on the regression line, leading to an overly perfect fit. As a result, the natural variability present in real-world data is not captured, causing the imputed dataset to exhibit biased correlations and reduced variance. Essentially, the data becomes too "clean," and this lack of variability can mislead subsequent analyses, making them overly optimistic or even erroneous.

Recognizing this limitation, stochastic regression imputation was introduced as an enhancement. Instead of merely using the fitted model, it adds a randomly drawn error term during the imputation process. This error term reintroduces the natural variability that the original regression imputation method missed. By doing so, the imputed values are scattered around the regression line, more accurately reflecting the true correlations and distributions in the dataset. This method, therefore, offers a more realistic representation of the data, ensuring that subsequent analyses are grounded in a dataset that mirrors genuine variability and relationships.

$Y \sim X + e$

$age \sim bmi + cholesterol + diastolicBP + error$

```r
imputation3 <- mice(NHANES17s, method = "norm.nob", # stochastic regression imputation
                    m = 1, maxit = 1, seed = 504, print = FALSE)

# look at all imputed values
imputation3$imp
#> $age
#>           1
#> 1  79.59513
#> 2  53.94601
#> 4  73.76486
#> 5  43.69817
#> 8  49.70112
#> 10 43.81002
#> 17 29.72590
#> 18 61.15172
#> 22 58.75506
#> 23 78.39545
#>
#> $bmi
#>           1
#> 8  27.53270
#> 18 31.52568
#>
#> $cholesterol
#>           1
#> 1  252.2928
```

```
#> 2  209.7680
#> 8  169.2450
#> 18 107.7585
#> 22 181.8617
#> 23 239.9008
#> 30 131.8489
#>
#> $diastolicBP
#>           1
#> 1  75.02181
#> 2  44.45935
#> 3  86.69637
#> 6  60.54256
#> 8  63.80884
#> 12 60.03311
#> 18 73.94575
#> 22 36.70323
#> 23 73.95647
#> 25 65.84012
#mice::complete(imputation3)

# examine the correlation between age and bmi before and after imputation
fit1 <- lm(age ~ bmi, NHANES17s)
summary(fit1)
#>
#> Call:
#> lm(formula = age ~ bmi, data = NHANES17s)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -37.383  -5.194   3.168   9.444  15.965
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  53.3482    17.1585   3.109  0.00606 **
#> bmi           0.1462     0.6063   0.241  0.81219
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 15.51 on 18 degrees of freedom
```

```r
#>   (10 observations deleted due to missingness)
#> Multiple R-squared:  0.003219,   Adjusted R-squared:  -0.05216
#> F-statistic: 0.05814 on 1 and 18 DF,  p-value: 0.8122
fit3 <- lm(age ~ bmi, mice::complete(imputation3))
summary(fit3)
#>
#> Call:
#> lm(formula = age ~ bmi, data = mice::complete(imputation3))
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -37.089  -6.691   3.183  10.104  21.288
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  60.4173    11.4671   5.269 1.33e-05 ***
#> bmi          -0.1206     0.4371  -0.276    0.785
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 15.53 on 28 degrees of freedom
#> Multiple R-squared:  0.002712,   Adjusted R-squared:  -0.03291
#> F-statistic: 0.07614 on 1 and 28 DF,  p-value: 0.7846
## Fitted coefficients of bmi are much closer before and after imputation
# with(data=NHANES17s, cor(age, bmi, use = "complete.obs"))

with(data=NHANES17s, cor(age, bmi, use = "pairwise.complete.obs"))
#> [1] 0.05674047
with(data = mice::complete(imputation3), cor(age, bmi))
#> [1] -0.05207502
# see the direction change?
```

0.0.0.0.4 * Predictive mean matching

Predictive Mean Matching (PMM) is an advanced imputation technique that aims to provide more realistic imputations for missing data. Let's break it down:

In this context, we're trying to fill in missing values for the variable 'age'. To do this, we use other variables like 'bmi',

'cholesterol', and 'diastolicBP' to predict 'age'. First, a regression model is run using the available data to estimate the relationship between 'age' and the predictor variables. From this model, we get a coefficient, which is then adjusted slightly to introduce some randomness. Using this adjusted coefficient, we predict the missing 'age' values for all subjects. For example, if 'subject 19' has a missing age value, we might predict it to be 45.5 years. Instead of using this predicted value directly, we look for other subjects who have actual age values and whose predicted ages are close to 45.5 years. From these subjects, one is randomly chosen, and their real age is used as the imputed value for 'subject 19'. In this way, PMM ensures that the imputed values are based on real, observed data from the dataset.

Tip

- Assume $Y =$ `age`, a variable with some missing values. X (say, `bmi`, `cholesterol`, `diastolicBP`) are predictors of Y.
- Estimate beta coef β from complete case running $Y \sim X + e$
- generate new $\beta* \sim Normal(b, se_b)$.
- using $\beta*$, predict new $\hat{Y}$ `predicted age` for all subjects (those with missing and observed `age`):
  - If `subject 19` (say) has missing values in `age` variable, find out his `predicted age` $\hat{Y}$ (say, 45.5).
  - Find others subjects, `subjects 2, 15, 24` (say) who has their `age` measured and their predicted age $\hat{Y}$ (say, `predicted ages` 43.9,45.7,46.1 with real ages 43,45,46 respectively) are close to `subject 19` (predicted age 45.5).
  - Randomly select `subject 2` with real/observed age 43, and impute 43 for `subject 19`'s missing age.

The strength of PMM lies in its approach. Instead of imputing a potentially artificial value based on a prediction, it im-

putes a real, observed value from the dataset. This ensures that the imputed data retains the original data's characteristics and doesn't introduce any unrealistic values. It offers a safeguard against extrapolation, ensuring that the imputed values are always within the plausible range of the dataset.

```
imputation3b <- mice(NHANES17s, method = "pmm",
                     m = 1, maxit = 1,
                     seed = 504, print = FALSE)
with(data=NHANES17s, cor(age, bmi, use = "pairwise.complete.obs"))
#> [1] 0.05674047
with(data = mice::complete(imputation3b), cor(age, bmi))
#> [1] -0.08029207
```

0.0.0.0.5 * Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Multiple imputation and workflow

- Ref: (Van Buuren 2018b), Sections 1.4 and 5.1
- Ref: (Buuren and Groothuis-Oudshoorn 2010)

We have learned different methods of imputation. In this section, we will introduce how to incorporate the data imputation into data analysis. In multiple imputation data analysis, three steps will be taken:

- **Step 0: Set imputation model**: Before starting the imputation process, it's crucial to determine the appropriate imputation model based on the nature of the missing data and the relationships between variables. This model will guide how the missing values are estimated. For instance, if the data is missing at random, a linear regression model might be used for continuous data, while logistic regression might be used for binary data. The choice of model

can significantly impact the quality of the imputed data, so it's essential to understand the underlying mechanisms causing the missingness and select a model accordingly.

- **Step 1: The incomplete dataset will be imputed** m times: In this step, the incomplete dataset is imputed multiple times, resulting in m different "complete" datasets. The reason for creating multiple datasets is to capture the uncertainty around the missing values. Each of these datasets will have slightly different imputed values, reflecting the variability and uncertainty in the imputation process. The number of imputations, m, is typically chosen based on the percentage of missing data and the desired level of accuracy. Common choices for m range from 5 to 50, but more imputations provide more accurate results, especially when the percentage of missing data is high.

- **Step 2: Each** m complete datasets will be analyzed separately by standard analysis (e.g., regression model): Once the m complete datasets are generated, each one is analyzed separately using standard statistical methods. For example, if the research question involves understanding the relationship between two variables, a regression model might be applied to each dataset. This step produces m sets of analysis results, one for each imputed dataset.

- **Step 3: The analysis results will be pooled / aggregated together by Rubin's rules (1987)**: The final step involves combining the results from the m separate analyses into a single set of results. This is done using Rubin's rules (1987) (R. Little and Rubin 1987), which provide a way to aggregate the estimates and adjust for the variability between the imputed datasets. The pooled results give a more accurate and robust estimate than analyzing a single imputed dataset. Rubin's rules ensure that the combined results reflect both the within-imputation variability (the variability in results from analyzing each dataset separately) and the between-imputation variability (the differences in results across the imputed datasets).

0.0.0.0.1 * Step 0

Set imputation model:

```
ini <- mice(data = NHANES17s, maxit = 0, print = FALSE)
pred <- ini$pred
pred
#>             age bmi cholesterol diastolicBP
#> age           0   1           1           1
#> bmi           1   0           1           1
#> cholesterol   1   1           0           1
#> diastolicBP   1   1           1           0
# A value of 1 indicates that column variables (say, bmi, cholesterol, diastolicBP)
# are used as a predictor to impute the a row variable (say, age).
pred[,"diastolicBP"] <- 0
# if you believe 'diastolicBP' should not be a predictor in any imputation model
pred
#>             age bmi cholesterol diastolicBP
#> age           0   1           1           0
#> bmi           1   0           1           0
#> cholesterol   1   1           0           0
#> diastolicBP   1   1           1           0
# for cholesterol: bmi and age used to predict cholesterol (diastolicBP is not a predictor)
# for diastolicBP: bmi, age and cholesterol used to predict diastolicBP
# (diastolicBP itself is not a predictor)
```

Set imputation method:

See Table 1 of (Van Buuren and Groothuis-Oudshoorn 2011).

```
meth <- ini$meth
meth
#>         age         bmi cholesterol diastolicBP
#>       "pmm"       "pmm"       "pmm"       "pmm"
# pmm is generally a good method,
# but let's see how to work with other methods
# just as an example.
# Specifying imputation method:
meth["bmi"] <- "mean"
# for BMI: no predictor used in mean method
# (only average of observed bmi)
```

```r
meth["cholesterol"] <- "norm.predict"
meth["diastolicBP"] <- "norm.nob"
meth
#>            age            bmi    cholesterol    diastolicBP
#>          "pmm"         "mean" "norm.predict"     "norm.nob"
```

Set imputation model based on correlation alone:

```r
predictor.selection <- quickpred(NHANES17s,
                                 mincor=0.1, # absolute correlation
                                 minpuc=0.1) # proportion of usable cases
predictor.selection
#>             age bmi cholesterol diastolicBP
#> age           0   1           1           1
#> bmi           0   0           0           0
#> cholesterol   1   1           0           1
#> diastolicBP   1   1           1           0
```

0.0.0.0.2 * Step 1

```r
# Step 1 Impute the incomplete data m=10 times
imputation4 <- mice(data=NHANES17s,
                    seed=504,
                    method = meth,
                    predictorMatrix = predictor.selection,
                    m=10, # imputation will be done 10 times (i.e., 10 imputed datasets)
                    maxit=3)
#>
#>  iter imp variable
#>   1   1  age  bmi  cholesterol  diastolicBP
#>   1   2  age  bmi  cholesterol  diastolicBP
#>   1   3  age  bmi  cholesterol  diastolicBP
#>   1   4  age  bmi  cholesterol  diastolicBP
#>   1   5  age  bmi  cholesterol  diastolicBP
#>   1   6  age  bmi  cholesterol  diastolicBP
#>   1   7  age  bmi  cholesterol  diastolicBP
#>   1   8  age  bmi  cholesterol  diastolicBP
#>   1   9  age  bmi  cholesterol  diastolicBP
#>   1   10  age  bmi  cholesterol  diastolicBP
```

```
#>   2   1  age  bmi  cholesterol  diastolicBP
#>   2   2  age  bmi  cholesterol  diastolicBP
#>   2   3  age  bmi  cholesterol  diastolicBP
#>   2   4  age  bmi  cholesterol  diastolicBP
#>   2   5  age  bmi  cholesterol  diastolicBP
#>   2   6  age  bmi  cholesterol  diastolicBP
#>   2   7  age  bmi  cholesterol  diastolicBP
#>   2   8  age  bmi  cholesterol  diastolicBP
#>   2   9  age  bmi  cholesterol  diastolicBP
#>   2   10  age  bmi  cholesterol  diastolicBP
#>   3   1  age  bmi  cholesterol  diastolicBP
#>   3   2  age  bmi  cholesterol  diastolicBP
#>   3   3  age  bmi  cholesterol  diastolicBP
#>   3   4  age  bmi  cholesterol  diastolicBP
#>   3   5  age  bmi  cholesterol  diastolicBP
#>   3   6  age  bmi  cholesterol  diastolicBP
#>   3   7  age  bmi  cholesterol  diastolicBP
#>   3   8  age  bmi  cholesterol  diastolicBP
#>   3   9  age  bmi  cholesterol  diastolicBP
#>   3   10  age  bmi  cholesterol  diastolicBP
imputation4$pred
#>             age bmi cholesterol diastolicBP
#> age           0   1           1           1
#> bmi           0   0           0           0
#> cholesterol   1   1           0           1
#> diastolicBP   1   1           1           0
## look at the variables used for imputation
mice::complete(imputation4, action = 1) # 1 imputed data
#>    age      bmi cholesterol diastolicBP
#> 1   70 17.50000    181.4426    83.04193
#> 2   72 15.70000    186.7818    74.75594
#> 3   66 31.70000    157.0000    76.19325
#> 4   75 21.50000    148.0000    74.00000
#> 5   66 18.10000    189.0000    38.00000
#> 6   66 23.70000    209.0000    60.21363
#> 7   75 38.90000    176.0000    66.00000
#> 8   66 25.12857    176.2946    79.70841
#> 9   56 21.30000    238.0000    68.00000
#> 10  70 19.70000    182.0000    68.00000
#> 11  67 23.50000    184.0000    70.00000
```

```
#> 12  54 39.90000    230.0000    70.54754
#> 13  71 22.50000    180.0000    60.00000
#> 14  61 30.70000    225.0000    72.00000
#> 15  22 24.50000    213.0000    62.00000
#> 16  45 22.00000    152.0000    88.00000
#> 17  67 26.00000     97.0000    62.00000
#> 18  66 25.12857    177.7121    68.60771
#> 19  60 35.90000    122.0000    68.00000
#> 20  60 23.80000    184.0000    68.00000
#> 21  64 22.40000    202.0000    72.00000
#> 22  66 14.70000    181.8733    52.65007
#> 23  66 16.10000    184.4578    65.22749
#> 24  67 31.10000    176.0000    52.00000
#> 25  70 23.90000    167.0000    59.15340
#> 26  53 33.40000    143.0000    74.00000
#> 27  42 27.60000    165.0000    86.00000
#> 28  57 28.60000    221.0000    74.00000
#> 29  20 27.60000    153.0000    54.00000
#> 30  72 21.30000    182.5833    76.00000
all <- mice::complete(imputation4, action="long") # combine all 5 imputed datasets
dim(all)
#> [1] 300   6
head(all)
#>   .imp .id age  bmi cholesterol diastolicBP
#> 1    1   1  70 17.5    181.4426    83.04193
#> 2    1   2  72 15.7    186.7818    74.75594
#> 3    1   3  66 31.7    157.0000    76.19325
#> 4    1   4  75 21.5    148.0000    74.00000
#> 5    1   5  66 18.1    189.0000    38.00000
#> 6    1   6  66 23.7    209.0000    60.21363
## you can change the way of displaying the data
data_hori <- mice::complete(imputation4, action="broad") # display five imputations horizontal
#> New names:
#> * `age` -> `age...1`
#> * `bmi` -> `bmi...2`
#> * `cholesterol` -> `cholesterol...3`
#> * `diastolicBP` -> `diastolicBP...4`
#> * `age` -> `age...5`
#> * `bmi` -> `bmi...6`
#> * `cholesterol` -> `cholesterol...7`
```

```
#> * `diastolicBP` -> `diastolicBP...8`
#> * `age` -> `age...9`
#> * `bmi` -> `bmi...10`
#> * `cholesterol` -> `cholesterol...11`
#> * `diastolicBP` -> `diastolicBP...12`
#> * `age` -> `age...13`
#> * `bmi` -> `bmi...14`
#> * `cholesterol` -> `cholesterol...15`
#> * `diastolicBP` -> `diastolicBP...16`
#> * `age` -> `age...17`
#> * `bmi` -> `bmi...18`
#> * `cholesterol` -> `cholesterol...19`
#> * `diastolicBP` -> `diastolicBP...20`
#> * `age` -> `age...21`
#> * `bmi` -> `bmi...22`
#> * `cholesterol` -> `cholesterol...23`
#> * `diastolicBP` -> `diastolicBP...24`
#> * `age` -> `age...25`
#> * `bmi` -> `bmi...26`
#> * `cholesterol` -> `cholesterol...27`
#> * `diastolicBP` -> `diastolicBP...28`
#> * `age` -> `age...29`
#> * `bmi` -> `bmi...30`
#> * `cholesterol` -> `cholesterol...31`
#> * `diastolicBP` -> `diastolicBP...32`
#> * `age` -> `age...33`
#> * `bmi` -> `bmi...34`
#> * `cholesterol` -> `cholesterol...35`
#> * `diastolicBP` -> `diastolicBP...36`
#> * `age` -> `age...37`
#> * `bmi` -> `bmi...38`
#> * `cholesterol` -> `cholesterol...39`
#> * `diastolicBP` -> `diastolicBP...40`

dim(data_hori)
#> [1] 30 40
head(data_hori)
#>   age.1 bmi.1 cholesterol.1 diastolicBP.1 age.2 bmi.2 cholesterol.2
#> 1    70  17.5      181.4426      83.04193    53  17.5      182.5155
#> 2    72  15.7      186.7818      74.75594    53  15.7      184.2733
```

```
#> 3    66  31.7      157.0000      76.19325    66  31.7      157.0000
#> 4    75  21.5      148.0000      74.00000    72  21.5      148.0000
#> 5    66  18.1      189.0000      38.00000    64  18.1      189.0000
#> 6    66  23.7      209.0000      60.21363    66  23.7      209.0000
#>   diastolicBP.2 age.3 bmi.3 cholesterol.3 diastolicBP.3 age.4 bmi.4
#> 1      46.75843    61  17.5      186.8438      57.28721    60  17.5
#> 2      63.10549    57  15.7      187.5062      58.23018    20  15.7
#> 3      68.71696    66  31.7      157.0000      60.52996    66  31.7
#> 4      74.00000    67  21.5      148.0000      74.00000    61  21.5
#> 5      38.00000    54  18.1      189.0000      38.00000    20  18.1
#> 6      70.44802    66  23.7      209.0000      84.12647    66  23.7
#>   cholesterol.4 diastolicBP.4 age.5 bmi.5 cholesterol.5 diastolicBP.5 age.6
#> 1      184.4672      55.46444    20  17.5      162.9816      55.71490    72
#> 2      184.5120      37.90855    22  15.7      166.5841      69.72937    56
#> 3      157.0000      75.44561    66  31.7      157.0000      59.98014    66
#> 4      148.0000      74.00000    22  21.5      148.0000      74.00000    75
#> 5      189.0000      38.00000    57  18.1      189.0000      38.00000    72
#> 6      209.0000      57.53259    66  23.7      209.0000      86.92689    66
#>   bmi.6 cholesterol.6 diastolicBP.6 age.7 bmi.7 cholesterol.7 diastolicBP.7
#> 1  17.5      187.0681      69.35847    75  17.5      180.3150      60.59233
#> 2  15.7      180.1763      68.08071    54  15.7      183.9833      91.04843
#> 3  31.7      157.0000      53.82798    66  31.7      157.0000      62.76216
#> 4  21.5      148.0000      74.00000    71  21.5      148.0000      74.00000
#> 5  18.1      189.0000      38.00000    54  18.1      189.0000      38.00000
#> 6  23.7      209.0000      76.06660    66  23.7      209.0000      63.70664
#>   age.8 bmi.8 cholesterol.8 diastolicBP.8 age.9 bmi.9 cholesterol.9
#> 1    42  17.5      182.0559      85.89369    71  17.5      181.8780
#> 2    57  15.7      183.0995      61.45652    20  15.7      190.0688
#> 3    66  31.7      157.0000      77.78881    66  31.7      157.0000
#> 4    61  21.5      148.0000      74.00000    60  21.5      148.0000
#> 5    57  18.1      189.0000      38.00000    70  18.1      189.0000
#> 6    66  23.7      209.0000      73.99950    66  23.7      209.0000
#>   diastolicBP.9 age.10 bmi.10 cholesterol.10 diastolicBP.10
#> 1      71.72672     67   17.5       184.5682       53.77248
#> 2      56.14070     22   15.7       182.6369       73.79396
#> 3      62.31531     66   31.7       157.0000       64.19636
#> 4      74.00000     64   21.5       148.0000       74.00000
#> 5      38.00000     71   18.1       189.0000       38.00000
#> 6      56.64013     66   23.7       209.0000       60.45355
```

```r
## Compare means of each imputed dataset
colMeans(data_hori)
#>          age.1          bmi.1  cholesterol.1  diastolicBP.1          age.2
#>       61.06667       25.12857      179.47152       68.06998       58.20000
#>          bmi.2  cholesterol.2  diastolicBP.2          age.3          bmi.3
#>       25.12857      179.71210       66.61289       57.40000       25.12857
#>  cholesterol.3  diastolicBP.3          age.4          bmi.4  cholesterol.4
#>      180.28681       68.77854       54.86667       25.12857      179.58873
#>  diastolicBP.4          age.5          bmi.5  cholesterol.5  diastolicBP.5
#>       66.13461       52.80000       25.12857      179.35478       66.74254
#>          age.6          bmi.6  cholesterol.6  diastolicBP.6          age.7
#>       60.00000       25.12857      179.22499       66.76592       58.43333
#>          bmi.7  cholesterol.7  diastolicBP.7          age.8          bmi.8
#>       25.12857      178.99900       66.82760       56.20000       25.12857
#>  cholesterol.8  diastolicBP.8          age.9          bmi.9  cholesterol.9
#>      179.63656       67.31160       59.06667       25.12857      179.58295
#>  diastolicBP.9         age.10         bmi.10 cholesterol.10 diastolicBP.10
#>       66.37078       55.63333       25.12857      179.69042       67.60832
```

0.0.0.0.3 * Step 2

```r
imputation4
#> Class: mids
#> Number of multiple imputations:  10
#> Imputation methods:
#>            age            bmi    cholesterol    diastolicBP
#>          "pmm"         "mean" "norm.predict"     "norm.nob"
#> PredictorMatrix:
#>             age bmi cholesterol diastolicBP
#> age           0   1           1           1
#> bmi           0   0           0           0
#> cholesterol   1   1           0           1
#> diastolicBP   1   1           1           0
```

```r
imputation4[[1]]
#>    age  bmi cholesterol diastolicBP
#> 1   NA 17.5          NA          NA
#> 2   NA 15.7          NA          NA
#> 3   66 31.7         157          NA
```

```
#> 4   NA 21.5          148           74
#> 5   NA 18.1          189           38
#> 6   66 23.7          209           NA
#> 7   75 38.9          176           66
#> 8   NA   NA           NA           NA
#> 9   56 21.3          238           68
#> 10  NA 19.7          182           68
#> 11  67 23.5          184           70
#> 12  54 39.9          230           NA
#> 13  71 22.5          180           60
#> 14  61 30.7          225           72
#> 15  22 24.5          213           62
#> 16  45 22.0          152           88
#> 17  NA 26.0           97           62
#> 18  NA   NA           NA           NA
#> 19  60 35.9          122           68
#> 20  60 23.8          184           68
#> 21  64 22.4          202           72
#> 22  NA 14.7           NA           NA
#> 23  NA 16.1           NA           NA
#> 24  67 31.1          176           52
#> 25  70 23.9          167           NA
#> 26  53 33.4          143           74
#> 27  42 27.6          165           86
#> 28  57 28.6          221           74
#> 29  20 27.6          153           54
#> 30  72 21.3           NA           76
```

```r
mice::complete(imputation4, action = 1)
#>    age      bmi cholesterol diastolicBP
#> 1   70 17.50000    181.4426    83.04193
#> 2   72 15.70000    186.7818    74.75594
#> 3   66 31.70000    157.0000    76.19325
#> 4   75 21.50000    148.0000    74.00000
#> 5   66 18.10000    189.0000    38.00000
#> 6   66 23.70000    209.0000    60.21363
#> 7   75 38.90000    176.0000    66.00000
#> 8   66 25.12857    176.2946    79.70841
#> 9   56 21.30000    238.0000    68.00000
#> 10  70 19.70000    182.0000    68.00000
```

```
#> 11  67 23.50000    184.0000    70.00000
#> 12  54 39.90000    230.0000    70.54754
#> 13  71 22.50000    180.0000    60.00000
#> 14  61 30.70000    225.0000    72.00000
#> 15  22 24.50000    213.0000    62.00000
#> 16  45 22.00000    152.0000    88.00000
#> 17  67 26.00000     97.0000    62.00000
#> 18  66 25.12857    177.7121    68.60771
#> 19  60 35.90000    122.0000    68.00000
#> 20  60 23.80000    184.0000    68.00000
#> 21  64 22.40000    202.0000    72.00000
#> 22  66 14.70000    181.8733    52.65007
#> 23  66 16.10000    184.4578    65.22749
#> 24  67 31.10000    176.0000    52.00000
#> 25  70 23.90000    167.0000    59.15340
#> 26  53 33.40000    143.0000    74.00000
#> 27  42 27.60000    165.0000    86.00000
#> 28  57 28.60000    221.0000    74.00000
#> 29  20 27.60000    153.0000    54.00000
#> 30  72 21.30000    182.5833    76.00000
```

```
mice::complete(imputation4, action = 10)
#>    age      bmi cholesterol diastolicBP
#> 1   67 17.50000    184.5682    53.77248
#> 2   22 15.70000    182.6369    73.79396
#> 3   66 31.70000    157.0000    64.19636
#> 4   64 21.50000    148.0000    74.00000
#> 5   71 18.10000    189.0000    38.00000
#> 6   66 23.70000    209.0000    60.45355
#> 7   75 38.90000    176.0000    66.00000
#> 8   71 25.12857    180.9696    81.81115
#> 9   56 21.30000    238.0000    68.00000
#> 10  22 19.70000    182.0000    68.00000
#> 11  67 23.50000    184.0000    70.00000
#> 12  54 39.90000    230.0000    78.04885
#> 13  71 22.50000    180.0000    60.00000
#> 14  61 30.70000    225.0000    72.00000
#> 15  22 24.50000    213.0000    62.00000
#> 16  45 22.00000    152.0000    88.00000
#> 17  56 26.00000     97.0000    62.00000
```

```
#> 18  57 25.12857    179.3458    85.12129
#> 19  60 35.90000    122.0000    68.00000
#> 20  60 23.80000    184.0000    68.00000
#> 21  64 22.40000    202.0000    72.00000
#> 22  20 14.70000    183.0146    57.66811
#> 23  71 16.10000    184.8780    62.76266
#> 24  67 31.10000    176.0000    52.00000
#> 25  70 23.90000    167.0000    58.62128
#> 26  53 33.40000    143.0000    74.00000
#> 27  42 27.60000    165.0000    86.00000
#> 28  57 28.60000    221.0000    74.00000
#> 29  20 27.60000    153.0000    54.00000
#> 30  72 21.30000    182.2995    76.00000
```

```r
# Step 2 Analyze the imputed data
fit4 <- with(data = imputation4, exp = lm(cholesterol ~ age + bmi + diastolicBP))
## fit model with each of 10 datasets separately
fit4
#> call :
#> with.mids(data = imputation4, expr = lm(cholesterol ~ age + bmi +
#>     diastolicBP))
#>
#> call1 :
#> mice(data = NHANES17s, m = 10, method = meth, predictorMatrix = predictor.selection,
#>     maxit = 3, seed = 504)
#>
#> nmis :
#>         age         bmi cholesterol diastolicBP
#>          10           2           7          10
#>
#> analyses :
#> [[1]]
#>
#> Call:
#> lm(formula = cholesterol ~ age + bmi + diastolicBP)
#>
#> Coefficients:
#> (Intercept)          age          bmi  diastolicBP
#>   210.68650     -0.19085     -0.52112     -0.09498
#>
```

```
#>
#> [[2]]
#>
#> Call:
#> lm(formula = cholesterol ~ age + bmi + diastolicBP)
#>
#> Coefficients:
#> (Intercept)          age          bmi  diastolicBP
#>   185.56395      0.06366     -0.49154      0.04196
#>
#>
#> [[3]]
#>
#> Call:
#> lm(formula = cholesterol ~ age + bmi + diastolicBP)
#>
#> Coefficients:
#> (Intercept)          age          bmi  diastolicBP
#>   188.01460     -0.07922     -0.52325      0.14493
#>
#>
#> [[4]]
#>
#> Call:
#> lm(formula = cholesterol ~ age + bmi + diastolicBP)
#>
#> Coefficients:
#> (Intercept)          age          bmi  diastolicBP
#>    210.2814       0.1096      -0.4602      -0.3802
#>
#>
#> [[5]]
#>
#> Call:
#> lm(formula = cholesterol ~ age + bmi + diastolicBP)
#>
#> Coefficients:
#> (Intercept)          age          bmi  diastolicBP
#>    167.8965       0.7171      -0.7613      -0.1090
#>
```

```
#>
#> [[6]]
#>
#> Call:
#> lm(formula = cholesterol ~ age + bmi + diastolicBP)
#>
#> Coefficients:
#> (Intercept)          age          bmi  diastolicBP
#>    187.4324       0.1727      -0.3452      -0.1482
#>
#>
#> [[7]]
#>
#> Call:
#> lm(formula = cholesterol ~ age + bmi + diastolicBP)
#>
#> Coefficients:
#> (Intercept)          age          bmi  diastolicBP
#>   203.45344     -0.22713     -0.37039     -0.02806
#>
#>
#> [[8]]
#>
#> Call:
#> lm(formula = cholesterol ~ age + bmi + diastolicBP)
#>
#> Coefficients:
#> (Intercept)          age          bmi  diastolicBP
#>  213.721570    -0.003491    -0.517870    -0.310132
#>
#>
#> [[9]]
#>
#> Call:
#> lm(formula = cholesterol ~ age + bmi + diastolicBP)
#>
#> Coefficients:
#> (Intercept)          age          bmi  diastolicBP
#>   205.74248     -0.16224     -0.46983     -0.07188
#>
```

```
#>
#> [[10]]
#>
#> Call:
#> lm(formula = cholesterol ~ age + bmi + diastolicBP)
#>
#> Coefficients:
#> (Intercept)          age          bmi  diastolicBP
#>    181.2751       0.0698      -0.5425       0.1208
```

0.0.0.0.4 * Step 3

0.0.0.0.4.1 * Understanding the pooled results

We will show the result of entire pool later. First we want to show the pooled results for the `age` variable only an an example.

```
require(dplyr)
res10 <- summary(fit4) %>% as_tibble %>% print(n=40)
#> # A tibble: 40 x 6
#>    term         estimate std.error statistic   p.value  nobs
#>    <chr>           <dbl>     <dbl>     <dbl>     <dbl> <int>
#>  1 (Intercept) 211.         53.6     3.93    0.000561     30
#>  2 age          -0.191       0.452  -0.422   0.676        30
#>  3 bmi          -0.521       0.943  -0.553   0.585        30
#>  4 diastolicBP  -0.0950      0.563  -0.169   0.867        30
#>  5 (Intercept) 186.         47.9     3.88    0.000644     30
#>  6 age           0.0637      0.459   0.139   0.891        30
#>  7 bmi          -0.492       0.939  -0.524   0.605        30
#>  8 diastolicBP   0.0420      0.533   0.0787  0.938        30
#>  9 (Intercept) 188.         48.9     3.85    0.000694     30
#> 10 age          -0.0792      0.470  -0.169   0.867        30
#> 11 bmi          -0.523       0.926  -0.565   0.577        30
#> 12 diastolicBP   0.145       0.547   0.265   0.793        30
#> 13 (Intercept) 210.         38.6     5.44    0.0000105    30
#> 14 age           0.110       0.372   0.295   0.771        30
#> 15 bmi          -0.460       0.950  -0.485   0.632        30
#> 16 diastolicBP  -0.380       0.534  -0.712   0.483        30
```

```
#> 17 (Intercept) 168.         39.6     4.24    0.000253     30
#> 18 age           0.717       0.299   2.39    0.0241       30
#> 19 bmi          -0.761       0.898  -0.848   0.404        30
#> 20 diastolicBP  -0.109       0.500  -0.218   0.829        30
#> 21 (Intercept) 187.         57.1     3.28    0.00294      30
#> 22 age           0.173       0.437   0.395   0.696        30
#> 23 bmi          -0.345       0.943  -0.366   0.717        30
#> 24 diastolicBP  -0.148       0.569  -0.260   0.797        30
#> 25 (Intercept) 203.         48.5     4.20    0.000278     30
#> 26 age          -0.227       0.390  -0.583   0.565        30
#> 27 bmi          -0.370       0.921  -0.402   0.691        30
#> 28 diastolicBP  -0.0281      0.536  -0.0523  0.959        30
#> 29 (Intercept) 214.         51.5     4.15    0.000313     30
#> 30 age          -0.00349     0.450  -0.00776 0.994        30
#> 31 bmi          -0.518       0.927  -0.559   0.581        30
#> 32 diastolicBP  -0.310       0.525  -0.590   0.560        30
#> 33 (Intercept) 206.         47.8     4.30    0.000213     30
#> 34 age          -0.162       0.392  -0.414   0.682        30
#> 35 bmi          -0.470       0.921  -0.510   0.614        30
#> 36 diastolicBP  -0.0719      0.523  -0.137   0.892        30
#> 37 (Intercept) 181.         44.4     4.08    0.000379     30
#> 38 age           0.0698      0.353   0.198   0.845        30
#> 39 bmi          -0.543       0.970  -0.559   0.581        30
#> 40 diastolicBP   0.121       0.558   0.216   0.830        30
m10 <- res10[res10$term == "age",]
m10
#> # A tibble: 10 x 6
#>    term  estimate std.error statistic p.value  nobs
#>    <chr>    <dbl>     <dbl>     <dbl>   <dbl> <int>
#>  1 age   -0.191       0.452  -0.422    0.676     30
#>  2 age    0.0637      0.459   0.139    0.891     30
#>  3 age   -0.0792      0.470  -0.169    0.867     30
#>  4 age    0.110       0.372   0.295    0.771     30
#>  5 age    0.717       0.299   2.39     0.0241    30
#>  6 age    0.173       0.437   0.395    0.696     30
#>  7 age   -0.227       0.390  -0.583    0.565     30
#>  8 age   -0.00349     0.450  -0.00776  0.994     30
#>  9 age   -0.162       0.392  -0.414    0.682     30
#> 10 age    0.0698      0.353   0.198    0.845     30
```

Let us describe the components of a pool for the `age` variable only:

```
m.number <- 10
# estimate = pooled estimate
# = sum of (m "beta-hat" estimates) / m (mean of m estimated statistics)
estimate <- mean(m10$estimate)
estimate
#> [1] 0.04699243
# ubar = sum of (m variance[beta] estimates) / m
# = within-imputation variance (mean of estimated variances)
ubar.var <- mean(m10$std.error^2)
ubar.var
#> [1] 0.1686837
# b =  variance of (m "beta-hat" estimates)
# = between-imputation variance
# (degree to which estimated statistic /
# "beta-hat" varies across m imputed datasets).
# This b is not available for single imputation when m = 1.
b.var <- var(m10$estimate)
b.var
#> [1] 0.07372796
# t = ubar + b + b/m = total variance according to Rubin's rules
# (within-imputation & between imputation variation)
t.var <- ubar.var + b.var + b.var/m.number
t.var
#> [1] 0.2497845
# riv = relative increase in variance
riv = (b.var + b.var/m.number)/ubar.var
riv
#> [1] 0.4807859
# lambda = proportion of variance to due nonresponse
lambda = (b.var + b.var/m.number)/t.var
lambda
#> [1] 0.3246829
# df (approximate for large sample without correction)
df.large.sample <- (m.number - 1)/lambda^2
df.large.sample
#> [1] 85.37359
# df (hypothetical complete data)
dfcom <- m10$nobs[1] - 4 # n = 30, # parameters = 4
```

```r
dfcom
#> [1] 26
# df (Barnard-Rubin correction)
df.obs <- (dfcom + 1)/(dfcom + 3) * dfcom * (1 - lambda)
df.c <- df.large.sample * df.obs/(df.large.sample + df.obs)
df.c
#> [1] 13.72019
# fmi = fraction of missing information per parameter
fmi = (riv + 2/(df.large.sample +3)) / (1 + riv)
fmi # based on large sample approximation
#> [1] 0.3399662
fmi = (riv + 2/(df.c +3)) / (1 + riv)
fmi # Barnard-Rubin correction
#> [1] 0.4054616
```

0.0.0.0.4.2 * Pooled estimate

Compare above results with the pooled table from `mice` below. Note that `df` is based on Barnard-Rubin correction and `fmi` value is calculated based on that corrected `df`.

```r
# Step 3 pool the analysis results
est1 <- mice::pool(fit4)
## pool all estimated together using Rubin's rule
est1
```

`Class: mipo  m = 10` (transposed version to accommodate space)

Term	(Intercept)	age	bmi	diastolicBP
m	10	10	10	10
Estimate	195.40679314	0.04699243	-0.50032666	-0.08347279
$\bar{u}$	2313.6362339	0.1686837	0.8722547	0.2909291
b	237.04075365	0.07372796	0.01274940	0.02857675
t	2574.3810629	0.2497845	0.8862790	0.3223635
df_com	26	26	26	26
df	21.22870	13.72019	23.80807	21.35356
RIV	0.11269915	0.48078595	0.01607826	0.10804843
λ	0.10128447	0.32468295	0.01582384	0.09751237

Term	(Intercept)	age	bmi	diastolicBP
FMI	0.17547051	0.40546159	0.08924771	0.17162783

Here:

- dfcom = df for complete data
- df = df with Barnard-Rubin correction

0.0.0.0.5 * Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Special case: Variable selection

Variable selection in analyzing missing data

- Ref: (Van Buuren 2018b), Section 5.4

The common workflow for analyzing missing data are (as mentioned above):

1. Imputing the data m times
2. Analyzing the m dataset
3. Pool all analysis together

We could apply variable selection in step 2, especially when we have no idea what is the best model to analyze the data. Howevere, it may become challenging when we pull all data together. With different dataset, the final model may or may not be the same.

We present the three method of variable selection on each imputed dataset presented by Buuren:

1. Majority: perform the model selection separately with m dataset and choose the variables that appears at least m/2 times
2. Stack: combine m datasets into a single dataset, and perform variable selection on this dataset
3. Wald (Rubin's rule): model selection was performed at model fitting step and combine the estimates using Rubin's rules. This is considered as gold standard.

0.0.0.0.1 * Majority using NHANES17s

```
data <- NHANES17s
imp <- mice(data, seed = 504, m = 100, print = FALSE)
## Multiple imputation with 100 imputations, resulting in 100 imputed datasets
scope0 <- list(upper = ~ age + bmi + cholesterol, lower = ~1)
expr <- expression(f1 <- lm(diastolicBP ~ age),
                   f2 <- step(f1, scope = scope0, trace = FALSE))
fit5 <- with(imp, expr)

## apply stepwise on each of the imputed dataset separately
formulas <- lapply(fit5$analyses, formula)
## fit5$analyses returns the selection result for each imputed dataset
terms <- lapply(formulas, terms)
votes <- unlist(lapply(terms, labels))
## look at the terms on each models
table(votes)
#> votes
#>         age         bmi cholesterol
#>           6          12           1
```

```
## Set up the stepwise variable selection, from null model to full model
scope <- list(upper = ~ age + bmi + cholesterol, lower = ~ age)

## Set up the stepwise variable selection, from important only model to full model
expr <- expression(f1 <- lm(diastolicBP ~ age),
                   f2 <- step(f1, scope = scope, trace = FALSE))
fit5 <- with(imp, expr)
## apply stepwise on each of the imputed dataset separately
formulas <- lapply(fit5$analyses, formula)
## fit5$analyses returns the selection result for each imputed dataset
```

```r
terms <- lapply(formulas, terms)
votes <- unlist(lapply(terms, labels))
## look at the terms on each models
table(votes)
#> votes
#>         age         bmi cholesterol
#>         100          11           1
```

0.0.0.0.2 * Stack using NHANES17s

```r
Stack.data <- mice::complete(imp, action="long")
head(Stack.data)
#>   .imp .id age  bmi cholesterol diastolicBP
#> 1    1   1  64 17.5         153          62
#> 2    1   2  67 15.7         152          60
#> 3    1   3  66 31.7         157          70
#> 4    1   4  67 21.5         148          74
#> 5    1   5  72 18.1         189          38
#> 6    1   6  66 23.7         209          54
tail(Stack.data)
#>      .imp .id age  bmi cholesterol diastolicBP
#> 2995  100  25  70 23.9         167          68
#> 2996  100  26  53 33.4         143          74
#> 2997  100  27  42 27.6         165          86
#> 2998  100  28  57 28.6         221          74
#> 2999  100  29  20 27.6         153          54
#> 3000  100  30  72 21.3         152          76
fitx <- lm(diastolicBP ~ age + bmi + cholesterol, data = Stack.data)
fity <- step(fitx, scope = scope0, trace = FALSE)
require(Publish)
#> Loading required package: Publish
#> Loading required package: prodlim
publish(fity)
#>     Variable Units Coefficient         CI.95 p-value
#>  (Intercept)             63.70 [62.12;65.28] < 1e-04
#>          bmi              0.14   [0.08;0.20] < 1e-04
```

0.0.0.0.3 * Wald using NHANES17s

```r
# m = 100
fit7 <- with(data=imp, expr=lm(diastolicBP ~ 1))
names(fit7)
#> [1] "call"     "call1"    "nmis"     "analyses"
fit7$analyses[[1]]
#>
#> Call:
#> lm(formula = diastolicBP ~ 1)
#>
#> Coefficients:
#> (Intercept)
#>       68.47
fit7$analyses[[100]]
#>
#> Call:
#> lm(formula = diastolicBP ~ 1)
#>
#> Coefficients:
#> (Intercept)
#>       65.47
fit8 <- with(data=imp, expr=lm(diastolicBP ~ bmi))
fit8$analyses[[45]]
#>
#> Call:
#> lm(formula = diastolicBP ~ bmi)
#>
#> Coefficients:
#> (Intercept)          bmi
#>    63.93092      0.09209
fit8$analyses[[99]]
#>
#> Call:
#> lm(formula = diastolicBP ~ bmi)
#>
#> Coefficients:
#> (Intercept)          bmi
#>    68.34740      0.01797
# The D1-statistics is the multivariate Wald test.
stat <- D1(fit8, fit7)
## use Wald test to see if we should add bmi into the model
```

```r
stat
#>    test statistic df1      df2 dfcom   p.value       riv
#>  1 ~~ 2 0.1215668   1 22.70437    28 0.7305539 0.5550013
# which indicates that adding bmi into our model might not be useful
```

```r
fit9 <- with(data=imp, expr=lm(diastolicBP ~ age + bmi))
stat <- D1(fit9, fit8)
## use Wald test to see if we should add age into the model
stat
#>    test   statistic df1      df2 dfcom   p.value       riv
#>  1 ~~ 2 0.006608523   1 22.46746    27 0.9359289 0.4545242
# which indicates that adding age into our model might not be useful
```

```r
fit10 <- with(data=imp, expr=lm(diastolicBP ~ age + bmi + cholesterol))
stat <- D1(fit10, fit9)
## use Wald test to see if we should add cholesterol into the model
stat
#>    test    statistic df1      df2 dfcom   p.value       riv
#>  1 ~~ 2 0.0003547819   1 22.14158    26 0.9851409 0.3615345
# which indicates that adding cholesterol into our model might not be useful
```

Try `method="likelihood"` as well.

```r
stat <- pool.compare(fit10, fit7, method = "likelihood", data=imp)
## test to see if we should add all 3 variables into the model
stat$pvalue
#> [1] 0.9432629
# which indicates that adding none of the variables into our model might be useful
```

0.0.0.0.4 * Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Assess the impact of missing values in model fitting

When working with datasets, missing values are a common challenge. These gaps in data can introduce biases and uncertainties, especially when we try to fit models to the data. To address this, researchers often use imputation methods to fill in the missing values based on the observed information. However, imputation itself can introduce uncertainties. Therefore, it's essential to assess the impact of these missing values on model fitting. Buuren, as referenced in (Van Buuren 2018b) Section 5.4.3, provides methods to do this. Out of the four methods presented by Buuren, the following two are the most commonly used:

1. Multiple imputation with more number of imputations (i.e., 200). Perform variable selection on each imputed dataset. The differences are attributed to the missing values
2. Bootstrapping the data from a single imputed dataset and do variable selection for each bootstrapping sample. We could evaluate sampling variation using this method

0.0.0.0.1 * Bootstrap using NHANES17s

```r
impx <- mice(NHANES17s, seed = 504, m=1, print=FALSE)
completedata <- mice::complete(impx)

set.seed(504)
votes <-c()
formula0 <- as.formula("diastolicBP ~ age + bmi + cholesterol")
scope <- list(upper = ~ age + bmi + cholesterol, lower = ~ age)

for (i in 1:200){
     ind <- sample(1:nrow(completedata),replace = TRUE)
     newdata <- completedata[ind,]
     full.model <- glm(formula0, data = newdata)
     f2 <- MASS::stepAIC(full.model,
                    scope = scope, trace = FALSE)
     formulas <- as.formula(f2)
     temp <- unlist(labels(terms(formulas)))
```

```r
    votes <- c(votes,temp)
 }
 table(votes)
#> votes
#>         age         bmi cholesterol
#>         200          59          17
 ## among 200 bootstrap samples how many times that each
 ## variable appears in the final model. Models have different
 ## variables are attributed to sampling variation
```

Convergence and diagnostics

Convergence

When an algorithm converges, it means that the sequence of estimates generated by the algorithm stabilizes and reaches a distribution that does not depend on the initial values. This is crucial for imputation and other statistical analyses because it indicates that the estimates are representative of the true underlying statistical properties and are not biased by initial assumptions or specific starting conditions.

- Ref: (Van Buuren 2018b), Section 6.5.2
- **MCMC Algorithm in MICE**: The MICE package implements a MCMC algorithm for imputation. The coefficients should be converged and irrelevant to the order which variable is imputed first.
- **Understanding pattern**: For convergence to be achieved, these chains should mix well with each other, meaning their paths should overlap and crisscross freely. If they show distinct, separate trends or paths, it indicates a lack of convergence, suggesting that the imputation may not be reliable.
- **Visualizing Convergence**: We could plot the imputation object to see the streams.

```r
## Recall the imputation we have done before
imputation5 <- mice(NHANES17s, seed = 504,
                    m=10,
                    maxit = 5,
                    print=FALSE)
plot(imputation5)
```

```r
## We hope to see no pattern in the trace lines
## Sometimes to comfirm this we may want to run with more iterations
imputation5_2 <- mice(NHANES17s, seed = 504,
                      m=10,
                      maxit = 50,
                      print=FALSE)
plot(imputation5_2)
```

Diagnostics

Model diagnostics plays a pivotal role in ensuring the robustness and accuracy of model fitting. Particularly in the realm of missing value imputations, where observed data serves as the foundation for estimating absent values, it becomes imperative to rigorously assess the imputation process. A straightforward diagnostic technique involves comparing the distributions of the observed data with the imputed values, especially when segmented or conditioned based on the variables that originally had missing entries. This comparison helps in discerning any discrepancies or biases introduced during the imputation, ensuring that the filled values align well with the inherent patterns of the observed data.

- Ref: (Van Buuren 2018b), Section 6.6

```r
## We could compare the imputed and observed data using Density plots
densityplot(imputation5, layout = c(2, 2))
```

```r
imputation5_3 <- mice(NHANES17s, seed = 504,
                      m=50,
                      maxit = 50,
                      print=FALSE)
densityplot(imputation5_3)
```

```r
## a subjective judgment on whether you think if there is significant discrepancy
bwplot(imputation5, age + bmi + cholesterol +diastolicBP ~ .imp, layout = c(2, 2))
```

```r
bwplot(imputation5_3)
```

```
## Plot a box plot to compare the imputed and observed values
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

References

Imputation in NHANES

This tutorial provides a comprehensive guide on handling and analyzing complex survey data with missing values. In analyzing complex survey data, a distinct approach is required compared to handling regular datasets. Specifically, the intricacies of survey design necessitate the consideration of primary sampling units/cluster, sampling weights, and stratification factors. These elements ensure that the analysis accurately reflects the survey's design and the underlying population structure. Recognizing and incorporating these factors is crucial for obtaining valid and representative insights from the data. As we delve into this tutorial, we'll explore how to effectively integrate these components into our missing data analysis process.

Complex Survey data

In the initial chunk, we load all the necessary libraries that will be used throughout the tutorial. These libraries provide functions and tools for data exploration, visualization, imputation, and analysis.

```r
# Load required packages
library(mice)
library(DataExplorer)
library(VIM)
library(jtools)
library(survey)
library(mitools)
```

Next, we load a dataset that contains survey data with some missing values. We then select specific columns or variables

from this dataset that we are interested in analyzing. To understand the extent and pattern of missingness in our data, we visualize it and display the missing data pattern.

```
load("Data/missingdata/NHANES17.RData")

Vars <- c("ID", "weight", "psu", "strata",
          "gender", "born", "race",
          "bmi", "cholesterol", "diabetes")
analyticx <- analytic.with.miss[,Vars]
plot_missing(analyticx)
```

```
md.pattern(analyticx)
```

```
#>       ID weight psu strata gender race born diabetes  bmi cholesterol
#> 6636  1      1   1      1      1    1    1        1    1           1    0
#> 1364  1      1   1      1      1    1    1        1    1           0    1
#> 97    1      1   1      1      1    1    1        1    0           1    1
#> 795   1      1   1      1      1    1    1        1    0           0    2
#> 4     1      1   1      1      1    1    1        0    1           1    1
#> 357   1      1   1      1      1    1    1        0    0           0    3
#> 1     1      1   1      1      1    1    0        1    1           1    1
#>       0      0   0      0      0    0    1      361 1249        2516 4127
```

Imputing

In the following chunk, we address the missing data by performing multiple imputations. This means that instead of filling in each missing value with a single estimate, we create multiple versions (datasets) where each missing value is filled in differently based on a specified algorithm. This helps in capturing the uncertainty around the missing values. The chunk sets up the parameters for multiple imputations, ensuring reproducibility and efficiency, and then performs the imputations on the dataset with missing values.

```
# imputation <- mice(analyticx, m=5, maxit=5, seed = 504007)
set.seed(504)
```

```r
imputation <- parlmice(analyticx, m=5, maxit=5, cluster.seed=504007)
#> Warning: 'parlmice' is deprecated.
#> Use 'futuremice' instead.
#> See help("Deprecated")
```

- **Data Input**: The primary input for the imputation function is the dataset with missing values. This dataset is what we aim to impute.
- **Number of Imputations**: The option `m=5` indicates that we want to create 5 different imputed datasets. Each of these datasets will have the missing values filled in slightly differently, based on the underlying imputation algorithm and the randomness introduced.
- **Maximum Iterations**: The imputation process is iterative, meaning it refines its estimates over several rounds. The option `maxit=5` specifies that the algorithm should run for a maximum of 5 iterations. This helps in achieving more accurate imputations, especially when the missing data mechanism is complex.
- **Setting Seed**: In computational processes that involve randomness, it's often useful to set a "seed" value. This ensures that the random processes are reproducible. If you run the imputation multiple times with the same seed, you'll get the same results each time. Two seed values are set in the chunk: one using the general `set.seed()` function and another specifically for the imputation function as `cluster.seed`.
- **Parallel Processing**: The function `parlmice` used for imputation indicates that the imputations are done in parallel. This means that instead of imputing one dataset after the other, the function tries to impute multiple datasets simultaneously (if the computational resources allow). This can speed up the process, especially when dealing with large datasets or when creating many imputed datasets.

Create new variable

After imputation, we might want to create new variables or modify existing ones to better suit our analysis. Here, we trans-

form one of the variables into a binary category based on a threshold. The choice of thresholds should be based on the data (i.e. median, quintiles, etc.) or on the literature (clinically relevant). This can help in simplifying the analysis or making the results more interpretable.

```r
impdata <- complete(imputation, action="long") #stacked data
impdata$cholesterol.bin <- ifelse(impdata$cholesterol < 200, "healthy", "unhealthy")
impdata$cholesterol.bin <- as.factor(impdata$cholesterol.bin)
dim(impdata)
#> [1] 46270    13
head(impdata)
#>   .imp .id    ID    weight psu strata gender                                born
#> 1    1   1 93703  8539.731   2    145 Female Born in 50 US states or Washingt
#> 2    1   2 93704 42566.615   1    143   Male Born in 50 US states or Washingt
#> 3    1   3 93705  8338.420   2    145 Female Born in 50 US states or Washingt
#> 4    1   4 93706  8723.440   2    134   Male Born in 50 US states or Washingt
#> 5    1   5 93707  7064.610   1    138   Male Born in 50 US states or Washingt
#> 6    1   6 93708 14372.489   2    138 Female                            Others
#>    race  bmi cholesterol diabetes cholesterol.bin
#> 1 Other 17.5         160       No         healthy
#> 2 White 15.7         186       No         healthy
#> 3 Black 31.7         157       No         healthy
#> 4 Other 21.5         148       No         healthy
#> 5 Other 18.1         189       No         healthy
#> 6 Other 23.7         209      Yes       unhealthy
```

Checking the data

After imputation, it's crucial to ensure that the imputed data maintains the integrity and structure of the original dataset. The following chunks are designed to help you visually and programmatically inspect the imputed data.

Visual Inspection of Missing Data:

In this chunk, we visually inspect the imputed datasets to see if there are any remaining missing values. We specifically look at the first two imputed datasets. Visualization tools like these can quickly show if the imputation process was successful in filling all missing values.

```r
plot_missing(subset(impdata, subset=.imp==1))
```

```r
plot_missing(subset(impdata, subset=.imp==2))
```

Comparing Original and Imputed Data (First Imputed Dataset):

- In this chunk, we focus on the first imputed dataset. We

extract this dataset and display the initial entries to get a sense of the data.

- We then remove any remaining missing values (if any) and display the initial entries of this complete dataset.
- Next, we compare the IDs (or unique identifiers) of the entries in the complete dataset with the original dataset to see which entries had missing values.
- We then create a new variable that indicates whether an entry had missing values or not and tabulate this information.

```r
analytic.miss1 <- subset(impdata, subset=.imp==1)
head(analytic.miss1$ID) # full data
#> [1] 93703 93704 93705 93706 93707 93708

analytic1 <- as.data.frame(na.omit(analytic.miss1))
head(analytic1$ID) # complete case
#> [1] 93703 93704 93705 93706 93707 93708

head(analytic.miss1$ID[analytic.miss1$ID %in% analytic1$ID])
#> [1] 93703 93704 93705 93706 93707 93708

analytic.miss1$miss <- 1
analytic.miss1$miss[analytic.miss1$ID %in% analytic1$ID] <- 0
table(analytic.miss1$miss)
#>
#>    0    1
#> 8892  362

head(analytic.miss1$ID[analytic.miss1$miss==1])
#> [1] 93710 93748 93786 93854 93865 93934
tail(analytic.miss1$ID[analytic.miss1$miss==1])
#> [1] 102840 102862 102919 102927 102928 102942
```

Comparing Original and Imputed Data (Second Imputed Dataset):

The this chunk is similar to the above but focuses on the second imputed dataset. We perform the same steps: extracting the dataset, removing missing values, comparing IDs, and creating a variable to indicate missingness.

```r
analytic.miss2 <- subset(impdata, subset=.imp==2)
head(analytic.miss2$ID) # full data
#> [1] 93703 93704 93705 93706 93707 93708

analytic2 <- as.data.frame(na.omit(analytic.miss2))
head(analytic2$ID) # complete case
#> [1] 93703 93704 93705 93706 93707 93708

head(analytic.miss2$ID[analytic.miss2$ID %in% analytic2$ID])
#> [1] 93703 93704 93705 93706 93707 93708

analytic.miss2$miss <- 1
analytic.miss2$miss[analytic.miss2$ID %in% analytic2$ID] <- 0
table(analytic.miss2$miss)
#>
#>    0    1
#> 8892  362

head(analytic.miss1$ID[analytic.miss1$miss==1])
#> [1] 93710 93748 93786 93854 93865 93934
tail(analytic.miss1$ID[analytic.miss1$miss==1])
#> [1] 102840 102862 102919 102927 102928 102942
```

Aggregating Missingness Information Across All Imputed Datasets:

- In the fourth chunk, we aim to consolidate the missingness information across all imputed datasets. We initialize a variable in the main dataset to indicate missingness.
- We then loop through each of the imputed datasets and update the main dataset's missingness variable based on the missingness information from each imputed dataset. This gives us a consolidated view of which entries had missing values across all imputed datasets.

```r
impdata$miss<-1
m <- 5
for (i in 1:m){
  impdata$miss[impdata$.imp == i] <- analytic.miss2$miss
  print(table(impdata$miss[impdata$.imp == i]))
```

```
}
#>
#>    0    1
#> 8892  362
#>
#>    0    1
#> 8892  362
#>
#>    0    1
#> 8892  362
#>
#>    0    1
#> 8892  362
#>
#>    0    1
#> 8892  362
```

Combining data

Since we have multiple versions of the imputed dataset, we need a way to combine them for analysis. In the next chunks, we use a method to merge these datasets into a single list, making it easier to apply subsequent analyses on all datasets simultaneously.

```
library(mitools)
allImputations <- imputationList(list(
  subset(impdata, subset=.imp==1),
  subset(impdata, subset=.imp==2),
  subset(impdata, subset=.imp==3),
  subset(impdata, subset=.imp==4),
  subset(impdata, subset=.imp==5)))
str(allImputations)
#> List of 2
#>  $ imputations:List of 5
#>   ..$ :'data.frame': 9254 obs. of  14 variables:
#>   .. ..$ .imp           : int [1:9254] 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ .id            : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
#>   .. ..$ ID             : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 9371
```

```
#>   .. ..$ weight         : num [1:9254] 8540 42567 8338 8723 7065 ...
#>   .. ..$ psu            : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
#>   .. ..$ strata         : Factor w/ 15 levels "134","135","136",..: 12 10 12 1 5 5 3 1 1 14
#>   .. ..$ gender         : chr [1:9254] "Female" "Male" "Female" "Male" ...
#>   .. ..$ born           : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st
#>   .. ..$ race           : chr [1:9254] "Other" "White" "Black" "Other" ...
#>   .. ..$ bmi            : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 18.3 21.3 19.7 ..
#>   .. ..$ cholesterol    : int [1:9254] 160 186 157 148 189 209 176 162 238 182 ...
#>   .. ..$ diabetes       : chr [1:9254] "No" "No" "No" "No" ...
#>   .. ..$ cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 1 2 1 ...
#>   .. ..$ miss           : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
#>   ..$ :'data.frame': 9254 obs. of  14 variables:
#>   .. ..$ .imp           : int [1:9254] 2 2 2 2 2 2 2 2 2 2 ...
#>   .. ..$ .id            : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
#>   .. ..$ ID             : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 9371
#>   .. ..$ weight         : num [1:9254] 8540 42567 8338 8723 7065 ...
#>   .. ..$ psu            : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
#>   .. ..$ strata         : Factor w/ 15 levels "134","135","136",..: 12 10 12 1 5 5 3 1 1 14
#>   .. ..$ gender         : chr [1:9254] "Female" "Male" "Female" "Male" ...
#>   .. ..$ born           : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st
#>   .. ..$ race           : chr [1:9254] "Other" "White" "Black" "Other" ...
#>   .. ..$ bmi            : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 44.8 21.3 19.7 ..
#>   .. ..$ cholesterol    : int [1:9254] 107 153 157 148 189 209 176 195 238 182 ...
#>   .. ..$ diabetes       : chr [1:9254] "No" "No" "No" "No" ...
#>   .. ..$ cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 1 2 1 ...
#>   .. ..$ miss           : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
#>   ..$ :'data.frame': 9254 obs. of  14 variables:
#>   .. ..$ .imp           : int [1:9254] 3 3 3 3 3 3 3 3 3 3 ...
#>   .. ..$ .id            : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
#>   .. ..$ ID             : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 9371
#>   .. ..$ weight         : num [1:9254] 8540 42567 8338 8723 7065 ...
#>   .. ..$ psu            : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
#>   .. ..$ strata         : Factor w/ 15 levels "134","135","136",..: 12 10 12 1 5 5 3 1 1 14
#>   .. ..$ gender         : chr [1:9254] "Female" "Male" "Female" "Male" ...
#>   .. ..$ born           : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st
#>   .. ..$ race           : chr [1:9254] "Other" "White" "Black" "Other" ...
#>   .. ..$ bmi            : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 13.6 21.3 19.7 ..
#>   .. ..$ cholesterol    : int [1:9254] 153 110 157 148 189 209 176 141 238 182 ...
#>   .. ..$ diabetes       : chr [1:9254] "No" "No" "No" "No" ...
#>   .. ..$ cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 1 2 1 ...
```

```
#>   .. ..$ miss           : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
#>   ..$ :'data.frame': 9254 obs. of  14 variables:
#>   .. ..$ .imp           : int [1:9254] 4 4 4 4 4 4 4 4 4 4 ...
#>   .. ..$ .id            : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
#>   .. ..$ ID             : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 9371
#>   .. ..$ weight         : num [1:9254] 8540 42567 8338 8723 7065 ...
#>   .. ..$ psu            : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
#>   .. ..$ strata         : Factor w/ 15 levels "134","135","136",..: 12 10 12 1 5 5 3 1 1 14
#>   .. ..$ gender         : chr [1:9254] "Female" "Male" "Female" "Male" ...
#>   .. ..$ born           : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st
#>   .. ..$ race           : chr [1:9254] "Other" "White" "Black" "Other" ...
#>   .. ..$ bmi            : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 19.2 21.3 19.7 ..
#>   .. ..$ cholesterol    : int [1:9254] 160 159 157 148 189 209 176 196 238 182 ...
#>   .. ..$ diabetes       : chr [1:9254] "No" "No" "No" "No" ...
#>   .. ..$ cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 1 2 1 ...
#>   .. ..$ miss           : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
#>   ..$ :'data.frame': 9254 obs. of  14 variables:
#>   .. ..$ .imp           : int [1:9254] 5 5 5 5 5 5 5 5 5 5 ...
#>   .. ..$ .id            : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
#>   .. ..$ ID             : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 9371
#>   .. ..$ weight         : num [1:9254] 8540 42567 8338 8723 7065 ...
#>   .. ..$ psu            : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
#>   .. ..$ strata         : Factor w/ 15 levels "134","135","136",..: 12 10 12 1 5 5 3 1 1 14
#>   .. ..$ gender         : chr [1:9254] "Female" "Male" "Female" "Male" ...
#>   .. ..$ born           : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st
#>   .. ..$ race           : chr [1:9254] "Other" "White" "Black" "Other" ...
#>   .. ..$ bmi            : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 26 21.3 19.7 ...
#>   .. ..$ cholesterol    : int [1:9254] 132 198 157 148 189 209 176 255 238 182 ...
#>   .. ..$ diabetes       : chr [1:9254] "No" "No" "No" "No" ...
#>   .. ..$ cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 2 2 1 ...
#>   .. ..$ miss           : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
#>  $ call       : language imputationList(list(subset(impdata, subset = .imp == 1), subset(im
#>  - attr(*, "class")= chr "imputationList"
```

Combining data efficiently

```r
m <- 5
set.seed(123)
```

```r
allImputations <-  imputationList(lapply(1:m,
                                         function(n)
                                           subset(impdata, subset=.imp==n)))
                                           #mice::complete(imputation, action = n)))
summary(allImputations)
#>             Length Class  Mode
#> imputations 5      -none- list
#> call        2      -none- call
str(allImputations)
#> List of 2
#>  $ imputations:List of 5
#>   ..$ :'data.frame': 9254 obs. of  14 variables:
#>   .. ..$ .imp           : int [1:9254] 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ .id            : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
#>   .. ..$ ID             : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 9371
#>   .. ..$ weight         : num [1:9254] 8540 42567 8338 8723 7065 ...
#>   .. ..$ psu            : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
#>   .. ..$ strata         : Factor w/ 15 levels "134","135","136",..: 12 10 12 1 5 5 3 1 1 14
#>   .. ..$ gender         : chr [1:9254] "Female" "Male" "Female" "Male" ...
#>   .. ..$ born           : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st
#>   .. ..$ race           : chr [1:9254] "Other" "White" "Black" "Other" ...
#>   .. ..$ bmi            : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 18.3 21.3 19.7 ..
#>   .. ..$ cholesterol    : int [1:9254] 160 186 157 148 189 209 176 162 238 182 ...
#>   .. ..$ diabetes       : chr [1:9254] "No" "No" "No" "No" ...
#>   .. ..$ cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 1 2 1 ...
#>   .. ..$ miss           : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
#>   ..$ :'data.frame': 9254 obs. of  14 variables:
#>   .. ..$ .imp           : int [1:9254] 2 2 2 2 2 2 2 2 2 2 ...
#>   .. ..$ .id            : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
#>   .. ..$ ID             : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 9371
#>   .. ..$ weight         : num [1:9254] 8540 42567 8338 8723 7065 ...
#>   .. ..$ psu            : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
#>   .. ..$ strata         : Factor w/ 15 levels "134","135","136",..: 12 10 12 1 5 5 3 1 1 14
#>   .. ..$ gender         : chr [1:9254] "Female" "Male" "Female" "Male" ...
#>   .. ..$ born           : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st
#>   .. ..$ race           : chr [1:9254] "Other" "White" "Black" "Other" ...
#>   .. ..$ bmi            : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 44.8 21.3 19.7 ..
#>   .. ..$ cholesterol    : int [1:9254] 107 153 157 148 189 209 176 195 238 182 ...
#>   .. ..$ diabetes       : chr [1:9254] "No" "No" "No" "No" ...
#>   .. ..$ cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 1 2 1 ...
```

```
#>   .. ..$ miss           : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
#>   ..$ :'data.frame': 9254 obs. of  14 variables:
#>   .. ..$ .imp           : int [1:9254] 3 3 3 3 3 3 3 3 3 3 ...
#>   .. ..$ .id            : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
#>   .. ..$ ID             : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 9371
#>   .. ..$ weight         : num [1:9254] 8540 42567 8338 8723 7065 ...
#>   .. ..$ psu            : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
#>   .. ..$ strata         : Factor w/ 15 levels "134","135","136",..: 12 10 12 1 5 5 3 1 1 14
#>   .. ..$ gender         : chr [1:9254] "Female" "Male" "Female" "Male" ...
#>   .. ..$ born           : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st
#>   .. ..$ race           : chr [1:9254] "Other" "White" "Black" "Other" ...
#>   .. ..$ bmi            : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 13.6 21.3 19.7 ..
#>   .. ..$ cholesterol    : int [1:9254] 153 110 157 148 189 209 176 141 238 182 ...
#>   .. ..$ diabetes       : chr [1:9254] "No" "No" "No" "No" ...
#>   .. ..$ cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 1 2 1 ...
#>   .. ..$ miss           : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
#>   ..$ :'data.frame': 9254 obs. of  14 variables:
#>   .. ..$ .imp           : int [1:9254] 4 4 4 4 4 4 4 4 4 4 ...
#>   .. ..$ .id            : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
#>   .. ..$ ID             : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 9371
#>   .. ..$ weight         : num [1:9254] 8540 42567 8338 8723 7065 ...
#>   .. ..$ psu            : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
#>   .. ..$ strata         : Factor w/ 15 levels "134","135","136",..: 12 10 12 1 5 5 3 1 1 14
#>   .. ..$ gender         : chr [1:9254] "Female" "Male" "Female" "Male" ...
#>   .. ..$ born           : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st
#>   .. ..$ race           : chr [1:9254] "Other" "White" "Black" "Other" ...
#>   .. ..$ bmi            : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 19.2 21.3 19.7 ..
#>   .. ..$ cholesterol    : int [1:9254] 160 159 157 148 189 209 176 196 238 182 ...
#>   .. ..$ diabetes       : chr [1:9254] "No" "No" "No" "No" ...
#>   .. ..$ cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 1 2 1 ...
#>   .. ..$ miss           : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
#>   ..$ :'data.frame': 9254 obs. of  14 variables:
#>   .. ..$ .imp           : int [1:9254] 5 5 5 5 5 5 5 5 5 5 ...
#>   .. ..$ .id            : int [1:9254] 1 2 3 4 5 6 7 8 9 10 ...
#>   .. ..$ ID             : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 9371
#>   .. ..$ weight         : num [1:9254] 8540 42567 8338 8723 7065 ...
#>   .. ..$ psu            : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
#>   .. ..$ strata         : Factor w/ 15 levels "134","135","136",..: 12 10 12 1 5 5 3 1 1 14
#>   .. ..$ gender         : chr [1:9254] "Female" "Male" "Female" "Male" ...
#>   .. ..$ born           : chr [1:9254] "Born in 50 US states or Washingt" "Born in 50 US st
```

```
#>   .. ..$ race           : chr [1:9254] "Other" "White" "Black" "Other" ...
#>   .. ..$ bmi            : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 26 21.3 19.7 ...
#>   .. ..$ cholesterol    : int [1:9254] 132 198 157 148 189 209 176 255 238 182 ...
#>   .. ..$ diabetes       : chr [1:9254] "No" "No" "No" "No" ...
#>   .. ..$ cholesterol.bin: Factor w/ 2 levels "healthy","unhealthy": 1 1 1 1 1 2 1 2 2 1 ...
#>   .. ..$ miss           : num [1:9254] 0 0 0 0 0 0 0 1 0 0 ...
#>  $ call       : language imputationList(lapply(1:m, function(n) subset(impdata, subset = .i
#>  - attr(*, "class")= chr "imputationList"
```

Logistic regression

With our imputed datasets ready, we proceed to fit a statistical model. Here, we use logistic regression as an example. We fit the model to each imputed dataset separately and then extract relevant statistics like odds ratios and confidence intervals.

```r
require(jtools)
require(survey)
data.list <- vector("list", m)
model.formula <- as.formula("I(cholesterol.bin=='healthy')~diabetes+gender+born+race+bmi")
```

```r
summary(allImputations$imputations[[1]]$weight)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>       0   12347   21060   34671   37562  419763
sum(allImputations$imputations[[1]]$weight==0)
#> [1] 550
w.design0 <- svydesign(ids=~psu, weights=~weight, strata=~strata,
                           data = allImputations, nest = TRUE)
w.design <- subset(w.design0, miss == 0)
fits <- with(w.design, svyglm(model.formula, family=quasibinomial))
#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(glm.object): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(glm.object): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(g): observations with zero weight not used for
```

```
#> calculating dispersion
#> Warning in summary.glm(glm.object): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(glm.object): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(glm.object): observations with zero weight not used for
#> calculating dispersion
# Estimate from first data
exp(coef(fits[[1]]))[2]
#> diabetesYes
#>    1.409246
exp(confint(fits[[1]]))[2,]
#>    2.5 %   97.5 %
#> 1.129997 1.757503
# Estimate from second data
exp(coef(fits[[2]]))[2]
#> diabetesYes
#>    1.437951
exp(confint(fits[[2]]))[2,]
#>    2.5 %   97.5 %
#> 1.171160 1.765518
```

Pooled / averaged estimates

After analyzing each imputed dataset separately, we need to combine the results to get a single set of estimates. This is done using a method that pools the results, taking into account the variability between the different imputed datasets.

```
pooled.estimates <- MIcombine(fits)
sum.pooled <- summary(pooled.estimates)
#> Multiple imputation results:
#>       with(w.design, svyglm(model.formula, family = quasibinomial))
#>       MIcombine.default(fits)
#>                    results          se        (lower       upper) missInfo
```

```
#> (Intercept)    2.13514754 0.236310332   1.667691110   2.60260397     19 %
#> diabetesYes    0.35209050 0.089197557   0.177261457   0.52691955      1 %
#> genderMale     0.17677460 0.088502097   0.003168621   0.35038058      5 %
#> bornOthers    -0.59642404 0.096782869  -0.786797572  -0.40605051     11 %
#> bornRefused  -12.33262819 0.708505945 -13.721274703 -10.94398169      0 %
#> raceHispanic   0.13913500 0.142910787  -0.141238429   0.41950842      6 %
#> raceOther     -0.17347602 0.105412185  -0.380201753   0.03324971      5 %
#> raceWhite     -0.35812104 0.134733737  -0.622240268  -0.09400181      2 %
#> bmi           -0.04011243 0.006523474  -0.053037557  -0.02718730     20 %
exp(sum.pooled[,1])
#> [1] 8.458294e+00 1.422037e+00 1.193362e+00 5.507777e-01 4.405626e-06
#> [6] 1.149279e+00 8.407373e-01 6.989885e-01 9.606814e-01
OR <- round(exp(pooled.estimates$coefficients),2)
OR <- as.data.frame(OR)
CI <- round(exp(confint(pooled.estimates)),2)
sig <- (CI[,1] < 1 & CI[,2] > 1)
sig <- ifelse(sig==FALSE, "*", "")
OR <- cbind(OR,CI,sig)
OR
#>                OR 2.5 % 97.5 % sig
#> (Intercept)  8.46  5.32  13.44   *
#> diabetesYes  1.42  1.19   1.69   *
#> genderMale   1.19  1.00   1.42   *
#> bornOthers   0.55  0.46   0.67   *
#> bornRefused  0.00  0.00   0.00   *
#> raceHispanic 1.15  0.87   1.52
#> raceOther    0.84  0.68   1.03
#> raceWhite    0.70  0.54   0.91   *
#> bmi          0.96  0.95   0.97   *
```

Step-by-step example

This segment offers a hands-on approach to understanding the imputation process. Here's a breakdown:

1. **Fitting Models to Individual Imputed Datasets**:
    - A list is initialized to store the results of models fitted to each imputed dataset.

- For every dataset, the specific imputed data is extracted.
- A survey design is established, considering factors like primary sampling units, stratification, and weights. This ensures the analysis aligns with the survey's design.
- This design is then refined to only consider complete data entries.
- A logistic regression model is then applied to this refined data.
- The results of this modeling are stored and displayed for review.

2. **Pooling Results from All Models**:

   - After individual analysis, the next step is to combine or 'pool' these results.
   - A special function is used to merge the results from all the models. This function accounts for variations between datasets and offers a combined estimate.
   - A summary of this combined data is then displayed, offering insights like coefficients, standard errors, and more. Another version of this summary, focusing on log-effects, is also presented for deeper insights.

```r
fits2 <- vector("list", m)
for (i in 1:m) {
  analytic.i <- allImputations$imputations[[i]]
  w.design0.i <- svydesign(id=~psu, strata=~strata, weights=~weight,
                           data=analytic.i, nest = TRUE)
  w.design.i <- subset(w.design0.i, miss == 0)
  fit <- svyglm(model.formula, design=w.design.i,
                family = quasibinomial("logit"))
  print(summ(fit))
  fits2[[i]] <- fit
}
#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(glm.object): observations with zero weight not used for
#> calculating dispersion
#> MODEL INFO:
```

```
#> Observations: 8892
#> Dependent Variable: I(cholesterol.bin == "healthy")
#> Type: Analysis of complex survey design
#>  Family: quasibinomial
#>  Link function: logit
#>
#> MODEL FIT:
#> Pseudo-R² (Cragg-Uhler) = 0.05
#> Pseudo-R² (McFadden) = 0.03
#> AIC =  NA
#>
#> ---------------------------------------------------
#>                        Est.   S.E.   t val.      p
#> ------------------ -------- ------ -------- ------
#> (Intercept)            2.26   0.20    11.32   0.00
#> diabetesYes            0.34   0.09     3.67   0.01
#> genderMale             0.17   0.09     1.88   0.10
#> bornOthers            -0.62   0.10    -6.36   0.00
#> bornRefused          -12.35   0.70   -17.63   0.00
#> raceHispanic           0.18   0.14     1.27   0.25
#> raceOther             -0.20   0.11    -1.77   0.12
#> raceWhite             -0.38   0.13    -2.83   0.03
#> bmi                   -0.04   0.01    -7.92   0.00
#> ---------------------------------------------------
#>
#> Estimated dispersion parameter = 0.99
#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion

#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion
#> MODEL INFO:
#> Observations: 8892
#> Dependent Variable: I(cholesterol.bin == "healthy")
#> Type: Analysis of complex survey design
#>  Family: quasibinomial
#>  Link function: logit
#>
#> MODEL FIT:
#> Pseudo-R² (Cragg-Uhler) = 0.05
```

```
#> Pseudo-R² (McFadden) = 0.03
#> AIC =  NA
#>
#> ---------------------------------------------------
#>                         Est.   S.E.   t val.      p
#> ------------------ -------- ------ -------- ------
#> (Intercept)             2.08   0.23     9.17   0.00
#> diabetesYes             0.36   0.09     4.19   0.00
#> genderMale              0.20   0.08     2.49   0.04
#> bornOthers             -0.63   0.08    -7.41   0.00
#> bornRefused           -12.33   0.71   -17.25   0.00
#> raceHispanic            0.13   0.14     0.95   0.37
#> raceOther              -0.16   0.10    -1.57   0.16
#> raceWhite              -0.37   0.14    -2.71   0.03
#> bmi                    -0.04   0.01    -6.05   0.00
#> ---------------------------------------------------
#>
#> Estimated dispersion parameter = 1
#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion

#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion
#> MODEL INFO:
#> Observations: 8892
#> Dependent Variable: I(cholesterol.bin == "healthy")
#> Type: Analysis of complex survey design
#>  Family: quasibinomial
#>  Link function: logit
#>
#> MODEL FIT:
#> Pseudo-R² (Cragg-Uhler) = 0.04
#> Pseudo-R² (McFadden) = 0.02
#> AIC =  NA
#>
#> ---------------------------------------------------
#>                         Est.   S.E.   t val.      p
#> ------------------ -------- ------ -------- ------
#> (Intercept)             2.03   0.22     9.22   0.00
#> diabetesYes             0.35   0.08     4.30   0.00
```

```
#> genderMale              0.17   0.09     1.88   0.10
#> bornOthers             -0.56   0.08    -7.21   0.00
#> bornRefused           -12.30   0.70   -17.49   0.00
#> raceHispanic            0.10   0.13     0.76   0.47
#> raceOther              -0.19   0.10    -1.81   0.11
#> raceWhite              -0.34   0.13    -2.59   0.04
#> bmi                    -0.04   0.01    -6.58   0.00
#> ----------------------------------------------------
#>
#> Estimated dispersion parameter = 0.99
#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion

#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion
#> MODEL INFO:
#> Observations: 8892
#> Dependent Variable: I(cholesterol.bin == "healthy")
#> Type: Analysis of complex survey design
#>  Family: quasibinomial
#>  Link function: logit
#>
#> MODEL FIT:
#> Pseudo-R² (Cragg-Uhler) = 0.05
#> Pseudo-R² (McFadden) = 0.03
#> AIC =  NA
#>
#> ----------------------------------------------------
#>                        Est.   S.E.   t val.      p
#> ------------------ -------- ------ -------- ------
#> (Intercept)             2.18   0.21    10.57   0.00
#> diabetesYes             0.36   0.09     3.75   0.01
#> genderMale              0.16   0.09     1.78   0.12
#> bornOthers             -0.57   0.10    -5.98   0.00
#> bornRefused           -12.35   0.71   -17.32   0.00
#> raceHispanic            0.12   0.14     0.83   0.43
#> raceOther              -0.16   0.10    -1.61   0.15
#> raceWhite              -0.34   0.13    -2.55   0.04
#> bmi                    -0.04   0.01    -7.11   0.00
#> ----------------------------------------------------
```

```
#>
#> Estimated dispersion parameter = 0.99
#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion

#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion
#> MODEL INFO:
#> Observations: 8892
#> Dependent Variable: I(cholesterol.bin == "healthy")
#> Type: Analysis of complex survey design
#>  Family: quasibinomial
#>  Link function: logit
#>
#> MODEL FIT:
#> Pseudo-R² (Cragg-Uhler) = 0.05
#> Pseudo-R² (McFadden) = 0.03
#> AIC =  NA
#>
#> ---------------------------------------------------
#>                        Est.   S.E.   t val.      p
#> ------------------ -------- ------ -------- ------
#> (Intercept)            2.12   0.22     9.67   0.00
#> diabetesYes            0.35   0.09     4.02   0.01
#> genderMale             0.19   0.08     2.30   0.06
#> bornOthers            -0.60   0.10    -6.06   0.00
#> bornRefused          -12.33   0.71   -17.40   0.00
#> raceHispanic           0.17   0.14     1.20   0.27
#> raceOther             -0.17   0.10    -1.64   0.14
#> raceWhite             -0.36   0.13    -2.78   0.03
#> bmi                   -0.04   0.01    -6.65   0.00
#> ---------------------------------------------------
#>
#> Estimated dispersion parameter = 0.99
```

```
pooled.estimates <- MIcombine(fits2)
summary(pooled.estimates)
#> Multiple imputation results:
#>       MIcombine.default(fits2)
#>                    results          se        (lower       upper) missInfo
```

```
#> (Intercept)    2.13514754 0.236310332   1.667691110   2.60260397     19 %
#> diabetesYes    0.35209050 0.089197557   0.177261457   0.52691955      1 %
#> genderMale     0.17677460 0.088502097   0.003168621   0.35038058      5 %
#> bornOthers    -0.59642404 0.096782869  -0.786797572  -0.40605051     11 %
#> bornRefused  -12.33262819 0.708505945 -13.721274703 -10.94398169      0 %
#> raceHispanic   0.13913500 0.142910787  -0.141238429   0.41950842      6 %
#> raceOther     -0.17347602 0.105412185  -0.380201753   0.03324971      5 %
#> raceWhite     -0.35812104 0.134733737  -0.622240268  -0.09400181      2 %
#> bmi           -0.04011243 0.006523474  -0.053037557  -0.02718730     20 %
summary(pooled.estimates,logeffect=TRUE, digits = 2)
#> Multiple imputation results:
#>       MIcombine.default(fits2)
#>              results      se  (lower  upper) missInfo
#> (Intercept)  8.5e+00 2.0e+00 5.3e+00 1.3e+01     19 %
#> diabetesYes  1.4e+00 1.3e-01 1.2e+00 1.7e+00      1 %
#> genderMale   1.2e+00 1.1e-01 1.0e+00 1.4e+00      5 %
#> bornOthers   5.5e-01 5.3e-02 4.6e-01 6.7e-01     11 %
#> bornRefused  4.4e-06 3.1e-06 1.1e-06 1.8e-05      0 %
#> raceHispanic 1.1e+00 1.6e-01 8.7e-01 1.5e+00      6 %
#> raceOther    8.4e-01 8.9e-02 6.8e-01 1.0e+00      5 %
#> raceWhite    7.0e-01 9.4e-02 5.4e-01 9.1e-01      2 %
#> bmi          9.6e-01 6.3e-03 9.5e-01 9.7e-01     20 %
```

Variable selection

Sometimes, not all variables in the dataset are relevant for our analysis. In the final chunks, we apply a method to select the most relevant variables for our model. This can help in simplifying the model and improving its interpretability.

```
require(jtools)
require(survey)
data.list <- vector("list", m)
model.formula <- as.formula("cholesterol~diabetes+gender+born+race+bmi")
scope <- list(upper = ~ diabetes+gender+born+race+bmi,
              lower = ~ diabetes)
for (i in 1:m) {
  analytic.i <- allImputations$imputations[[i]]
  w.design0.i <- svydesign(id=~psu, strata=~strata, weights=~weight,
```

```r
                        data=analytic.i, nest = TRUE)
  w.design.i <- subset(w.design0.i, miss == 0)
  fit <- svyglm(model.formula, design=w.design.i)
  fitstep <- step(fit, scope = scope, trace = FALSE,
                              direction = "backward")
  data.list[[i]] <- fitstep
}
#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(glm.object): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(glm.object): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(glm.object): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(glm.object): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(g): observations with zero weight not used for
#> calculating dispersion
#> Warning in summary.glm(glm.object): observations with zero weight not used for
#> calculating dispersion
```

Check out the variables selected

```r
x <- all.vars(formula(fit))
for (i in 1:m) x <- c(x, all.vars(formula(data.list[[i]])))
table(x)-1
#> x
#>         bmi        born cholesterol    diabetes      gender        race
#>           5           5           5           5           5           5
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Missing in outcome

This section provides a theoretical background on the concept of Multiple Imputation and then Deletion (MID). It highlights the challenges of imputing dependent and exposure variables and introduces the idea of using auxiliary variables to aid imputation. The section also contrasts the results of traditional MI with MID, especially when the number of imputed datasets is high.

- Often researchers are reluctant to impute values in the **dependent variable** (and exposure variable). Particularly, for dependent variable, imputation might not help too much.
- However, if you have a good **auxiliary variable** (e.g., strongly correlated predictor, that are not used in the main analysis), often multiple imputation method can help. Use of auxiliary variables is one of the greatest strengths of MI methods.
- MI algorithm generally do not have any special treatment for dependent variable in its original form, and hence **ignoring dependent variable completely** may not be a good idea in many scenarios.
- Multiple imputation followed by deletion of imputed outcomes is known as **MID**. This is very popular, especially when you have high percentage missing values in the outcome variable (e.g., 20%-50%). For low missing % in outcome, the advantage can be minimal.
- We are extending this idea to deletion of **imputed exposures** as well (researchers are often reluctant to impute primary exposure of interest).
- Original MI and MID may result in similar results when m (**number of imputed datasets**) is higher.

Data

In the initial chunk, we load several packages that provide functions and tools necessary for the subsequent analysis. These packages facilitate multiple imputation, data visualization, and statistical modeling among other tasks.

```r
# Load required packages
library(mice)
library(DataExplorer)
library(VIM)
library(jtools)
library(survey)
library(mitools)
```

We load the necessary data:

```r
load("Data/missingdata/NHANES17.RData")
```

The data is briefly inspected to understand its structure. An identifier column is added to uniquely identify each row or observation in the dataset.

```r
require(mice)
nhanes2
#>      age  bmi  hyp chl
#> 1  20-39   NA <NA>  NA
#> 2  40-59 22.7   no 187
#> 3  20-39   NA   no 187
#> 4  60-99   NA <NA>  NA
#> 5  20-39 20.4   no 113
#> 6  60-99   NA <NA> 184
#> 7  20-39 22.5   no 118
#> 8  20-39 30.1   no 187
#> 9  40-59 22.0   no 238
#> 10 40-59   NA <NA>  NA
#> 11 20-39   NA <NA>  NA
#> 12 40-59   NA <NA>  NA
#> 13 60-99 21.7   no 206
#> 14 40-59 28.7  yes 204
```

```
#> 15 20-39 29.6   no  NA
#> 16 20-39   NA <NA>  NA
#> 17 60-99 27.2  yes 284
#> 18 40-59 26.3  yes 199
#> 19 20-39 35.3   no 218
#> 20 60-99 25.5  yes  NA
#> 21 20-39   NA <NA>  NA
#> 22 20-39 33.2   no 229
#> 23 20-39 27.5   no 131
#> 24 60-99 24.9   no  NA
#> 25 40-59 27.4   no 186
nhanes2$id <- 1:nrow(nhanes2)
nhanes2
#>      age  bmi  hyp chl id
#> 1  20-39   NA <NA>  NA  1
#> 2  40-59 22.7   no 187  2
#> 3  20-39   NA   no 187  3
#> 4  60-99   NA <NA>  NA  4
#> 5  20-39 20.4   no 113  5
#> 6  60-99   NA <NA> 184  6
#> 7  20-39 22.5   no 118  7
#> 8  20-39 30.1   no 187  8
#> 9  40-59 22.0   no 238  9
#> 10 40-59   NA <NA>  NA 10
#> 11 20-39   NA <NA>  NA 11
#> 12 40-59   NA <NA>  NA 12
#> 13 60-99 21.7   no 206 13
#> 14 40-59 28.7  yes 204 14
#> 15 20-39 29.6   no  NA 15
#> 16 20-39   NA <NA>  NA 16
#> 17 60-99 27.2  yes 284 17
#> 18 40-59 26.3  yes 199 18
#> 19 20-39 35.3   no 218 19
#> 20 60-99 25.5  yes  NA 20
#> 21 20-39   NA <NA>  NA 21
#> 22 20-39 33.2   no 229 22
#> 23 20-39 27.5   no 131 23
#> 24 60-99 24.9   no  NA 24
#> 25 40-59 27.4   no 186 25
```

Outcome and exposure has missing

This chunk focuses on identifying which rows have missing values in both the outcome and exposure variables. The outcome and exposure variables are crucial for the analysis, so understanding where they are missing is essential.

```
# assume outcome = bmi and exposure = chl
nhanes2.excludingYA <- subset(nhanes2, !is.na(bmi) & !is.na(chl) )
nhanes2.excludingYA # data without missing A and Y
#>      age  bmi hyp chl id
#> 2  40-59 22.7  no 187  2
#> 5  20-39 20.4  no 113  5
#> 7  20-39 22.5  no 118  7
#> 8  20-39 30.1  no 187  8
#> 9  40-59 22.0  no 238  9
#> 13 60-99 21.7  no 206 13
#> 14 40-59 28.7 yes 204 14
#> 17 60-99 27.2 yes 284 17
#> 18 40-59 26.3 yes 199 18
#> 19 20-39 35.3  no 218 19
#> 22 20-39 33.2  no 229 22
#> 23 20-39 27.5  no 131 23
#> 25 40-59 27.4  no 186 25
# identify ids of subjects with missing A & Y
nhanes2.excludingYA$id
#>  [1]  2  5  7  8  9 13 14 17 18 19 22 23 25
```

Impute as usual

Using the entire dataset, missing values are imputed. This is done by first initializing an imputation model and then performing the imputation to create multiple datasets where missing values are filled in. The result is a list of datasets with imputed values. That means, we impute Y and A for now, as well as other covariates with missing values.

```
# use full data to impute
ini <- mice(nhanes2, pri = FALSE)
ini$method
```

```
#>      age      bmi      hyp      chl       id
#>       ""    "pmm" "logreg"    "pmm"       ""
pred <- ini$predictorMatrix
pred
#>     age bmi hyp chl id
#> age   0   1   1   1  1
#> bmi   1   0   1   1  1
#> hyp   1   1   0   1  1
#> chl   1   1   1   0  1
#> id    1   1   1   1  0
pred[,"id"] <- 0 # as this is not a predictor
m <- 5
imp <- mice(data=nhanes2, m=m, maxit=3, seed=504007)
#>
#>  iter imp variable
#>   1   1  bmi  hyp  chl
#>   1   2  bmi  hyp  chl
#>   1   3  bmi  hyp  chl
#>   1   4  bmi  hyp  chl
#>   1   5  bmi  hyp  chl
#>   2   1  bmi  hyp  chl
#>   2   2  bmi  hyp  chl
#>   2   3  bmi  hyp  chl
#>   2   4  bmi  hyp  chl
#>   2   5  bmi  hyp  chl
#>   3   1  bmi  hyp  chl
#>   3   2  bmi  hyp  chl
#>   3   3  bmi  hyp  chl
#>   3   4  bmi  hyp  chl
#>   3   5  bmi  hyp  chl
# list format in m data
impdata <- mice::complete(imp, action = "all")
impdata # all IDs are present
#> $`1`
#>      age  bmi hyp chl id
#> 1  20-39 20.4  no 187  1
#> 2  40-59 22.7  no 187  2
#> 3  20-39 27.4  no 187  3
#> 4  60-99 25.5 yes 204  4
#> 5  20-39 20.4  no 113  5
```

```
#> 6  60-99 22.5 yes 184  6
#> 7  20-39 22.5  no 118  7
#> 8  20-39 30.1  no 187  8
#> 9  40-59 22.0  no 238  9
#> 10 40-59 20.4  no 199 10
#> 11 20-39 27.5  no 199 11
#> 12 40-59 26.3  no 284 12
#> 13 60-99 21.7  no 206 13
#> 14 40-59 28.7 yes 204 14
#> 15 20-39 29.6  no 187 15
#> 16 20-39 35.3 yes 204 16
#> 17 60-99 27.2 yes 284 17
#> 18 40-59 26.3 yes 199 18
#> 19 20-39 35.3  no 218 19
#> 20 60-99 25.5 yes 284 20
#> 21 20-39 35.3 yes 184 21
#> 22 20-39 33.2  no 229 22
#> 23 20-39 27.5  no 131 23
#> 24 60-99 24.9  no 218 24
#> 25 40-59 27.4  no 186 25
#>
#> $`2`
#>      age  bmi hyp chl id
#> 1  20-39 28.7 yes 238  1
#> 2  40-59 22.7  no 187  2
#> 3  20-39 30.1  no 187  3
#> 4  60-99 22.5 yes 218  4
#> 5  20-39 20.4  no 113  5
#> 6  60-99 20.4 yes 184  6
#> 7  20-39 22.5  no 118  7
#> 8  20-39 30.1  no 187  8
#> 9  40-59 22.0  no 238  9
#> 10 40-59 22.5 yes 238 10
#> 11 20-39 26.3 yes 118 11
#> 12 40-59 20.4  no 199 12
#> 13 60-99 21.7  no 206 13
#> 14 40-59 28.7 yes 204 14
#> 15 20-39 29.6  no 187 15
#> 16 20-39 24.9  no 238 16
#> 17 60-99 27.2 yes 284 17
```

```
#> 18 40-59 26.3 yes 199 18
#> 19 20-39 35.3  no 218 19
#> 20 60-99 25.5 yes 218 20
#> 21 20-39 27.5  no 187 21
#> 22 20-39 33.2  no 229 22
#> 23 20-39 27.5  no 131 23
#> 24 60-99 24.9  no 218 24
#> 25 40-59 27.4  no 186 25
#>
#> $`3`
#>      age  bmi hyp chl id
#> 1  20-39 25.5  no 229  1
#> 2  40-59 22.7  no 187  2
#> 3  20-39 22.0  no 187  3
#> 4  60-99 20.4  no 199  4
#> 5  20-39 20.4  no 113  5
#> 6  60-99 22.7  no 184  6
#> 7  20-39 22.5  no 118  7
#> 8  20-39 30.1  no 187  8
#> 9  40-59 22.0  no 238  9
#> 10 40-59 27.4  no 184 10
#> 11 20-39 30.1  no 238 11
#> 12 40-59 22.0  no 186 12
#> 13 60-99 21.7  no 206 13
#> 14 40-59 28.7 yes 204 14
#> 15 20-39 29.6  no 229 15
#> 16 20-39 22.0  no 118 16
#> 17 60-99 27.2 yes 284 17
#> 18 40-59 26.3 yes 199 18
#> 19 20-39 35.3  no 218 19
#> 20 60-99 25.5 yes 218 20
#> 21 20-39 29.6  no 131 21
#> 22 20-39 33.2  no 229 22
#> 23 20-39 27.5  no 131 23
#> 24 60-99 24.9  no 218 24
#> 25 40-59 27.4  no 186 25
#>
#> $`4`
#>      age  bmi hyp chl id
#> 1  20-39 24.9  no 187  1
```

```
#> 2  40-59 22.7  no 187  2
#> 3  20-39 27.4  no 187  3
#> 4  60-99 24.9 yes 206  4
#> 5  20-39 20.4  no 113  5
#> 6  60-99 22.5  no 184  6
#> 7  20-39 22.5  no 118  7
#> 8  20-39 30.1  no 187  8
#> 9  40-59 22.0  no 238  9
#> 10 40-59 26.3 yes 187 10
#> 11 20-39 29.6  no 229 11
#> 12 40-59 27.4  no 204 12
#> 13 60-99 21.7  no 206 13
#> 14 40-59 28.7 yes 204 14
#> 15 20-39 29.6  no 186 15
#> 16 20-39 35.3  no 184 16
#> 17 60-99 27.2 yes 284 17
#> 18 40-59 26.3 yes 199 18
#> 19 20-39 35.3  no 218 19
#> 20 60-99 25.5 yes 199 20
#> 21 20-39 27.5  no 187 21
#> 22 20-39 33.2  no 229 22
#> 23 20-39 27.5  no 131 23
#> 24 60-99 24.9  no 284 24
#> 25 40-59 27.4  no 186 25
#>
#> $`5`
#>      age  bmi hyp chl id
#> 1  20-39 27.2  no 238  1
#> 2  40-59 22.7  no 187  2
#> 3  20-39 24.9  no 187  3
#> 4  60-99 20.4  no 229  4
#> 5  20-39 20.4  no 113  5
#> 6  60-99 21.7  no 184  6
#> 7  20-39 22.5  no 118  7
#> 8  20-39 30.1  no 187  8
#> 9  40-59 22.0  no 238  9
#> 10 40-59 20.4 yes 187 10
#> 11 20-39 25.5  no 118 11
#> 12 40-59 21.7  no 187 12
#> 13 60-99 21.7  no 206 13
```

```
#> 14 40-59 28.7 yes 204 14
#> 15 20-39 29.6  no 199 15
#> 16 20-39 27.5  no 187 16
#> 17 60-99 27.2 yes 284 17
#> 18 40-59 26.3 yes 199 18
#> 19 20-39 35.3  no 218 19
#> 20 60-99 25.5 yes 206 20
#> 21 20-39 33.2  no 206 21
#> 22 20-39 33.2  no 229 22
#> 23 20-39 27.5  no 131 23
#> 24 60-99 24.9  no 204 24
#> 25 40-59 27.4  no 186 25
#>
#> attr(,"class")
#> [1] "mild" "list"
```

Include a missing indicator (Y & A)

For each imputed dataset, a new column is added to indicate whether the outcome and exposure variables were originally missing. This "missing indicator" column will be used later to subset the data.

```
# Define formula (making binary Y)
formula <- as.formula("I(bmi>25) ~ chl + hyp")
data.list <- vector("list", m)
# subset the data without Y and A's that had missing values
# and record those subset data
for (i in 1:m) {
  analytic.i <- impdata[[i]]
  analytic.i$miss <- 1
  analytic.i$miss[analytic.i$id %in% nhanes2.excludingYA$id] <- 0
  data.list[[i]] <- analytic.i
}
data.list  # only relevant IDs are present
#> [[1]]
#>      age  bmi hyp chl id miss
#> 1  20-39 20.4  no 187  1    1
#> 2  40-59 22.7  no 187  2    0
```

```
#> 3  20-39 27.4  no 187  3    1
#> 4  60-99 25.5 yes 204  4    1
#> 5  20-39 20.4  no 113  5    0
#> 6  60-99 22.5 yes 184  6    1
#> 7  20-39 22.5  no 118  7    0
#> 8  20-39 30.1  no 187  8    0
#> 9  40-59 22.0  no 238  9    0
#> 10 40-59 20.4  no 199 10    1
#> 11 20-39 27.5  no 199 11    1
#> 12 40-59 26.3  no 284 12    1
#> 13 60-99 21.7  no 206 13    0
#> 14 40-59 28.7 yes 204 14    0
#> 15 20-39 29.6  no 187 15    1
#> 16 20-39 35.3 yes 204 16    1
#> 17 60-99 27.2 yes 284 17    0
#> 18 40-59 26.3 yes 199 18    0
#> 19 20-39 35.3  no 218 19    0
#> 20 60-99 25.5 yes 284 20    1
#> 21 20-39 35.3 yes 184 21    1
#> 22 20-39 33.2  no 229 22    0
#> 23 20-39 27.5  no 131 23    0
#> 24 60-99 24.9  no 218 24    1
#> 25 40-59 27.4  no 186 25    0
#>
#> [[2]]
#>      age  bmi hyp chl id miss
#> 1  20-39 28.7 yes 238  1    1
#> 2  40-59 22.7  no 187  2    0
#> 3  20-39 30.1  no 187  3    1
#> 4  60-99 22.5 yes 218  4    1
#> 5  20-39 20.4  no 113  5    0
#> 6  60-99 20.4 yes 184  6    1
#> 7  20-39 22.5  no 118  7    0
#> 8  20-39 30.1  no 187  8    0
#> 9  40-59 22.0  no 238  9    0
#> 10 40-59 22.5 yes 238 10    1
#> 11 20-39 26.3 yes 118 11    1
#> 12 40-59 20.4  no 199 12    1
#> 13 60-99 21.7  no 206 13    0
#> 14 40-59 28.7 yes 204 14    0
```

```
#> 15 20-39 29.6  no 187 15    1
#> 16 20-39 24.9  no 238 16    1
#> 17 60-99 27.2 yes 284 17    0
#> 18 40-59 26.3 yes 199 18    0
#> 19 20-39 35.3  no 218 19    0
#> 20 60-99 25.5 yes 218 20    1
#> 21 20-39 27.5  no 187 21    1
#> 22 20-39 33.2  no 229 22    0
#> 23 20-39 27.5  no 131 23    0
#> 24 60-99 24.9  no 218 24    1
#> 25 40-59 27.4  no 186 25    0
#>
#> [[3]]
#>      age  bmi hyp chl id miss
#> 1  20-39 25.5  no 229  1    1
#> 2  40-59 22.7  no 187  2    0
#> 3  20-39 22.0  no 187  3    1
#> 4  60-99 20.4  no 199  4    1
#> 5  20-39 20.4  no 113  5    0
#> 6  60-99 22.7  no 184  6    1
#> 7  20-39 22.5  no 118  7    0
#> 8  20-39 30.1  no 187  8    0
#> 9  40-59 22.0  no 238  9    0
#> 10 40-59 27.4  no 184 10    1
#> 11 20-39 30.1  no 238 11    1
#> 12 40-59 22.0  no 186 12    1
#> 13 60-99 21.7  no 206 13    0
#> 14 40-59 28.7 yes 204 14    0
#> 15 20-39 29.6  no 229 15    1
#> 16 20-39 22.0  no 118 16    1
#> 17 60-99 27.2 yes 284 17    0
#> 18 40-59 26.3 yes 199 18    0
#> 19 20-39 35.3  no 218 19    0
#> 20 60-99 25.5 yes 218 20    1
#> 21 20-39 29.6  no 131 21    1
#> 22 20-39 33.2  no 229 22    0
#> 23 20-39 27.5  no 131 23    0
#> 24 60-99 24.9  no 218 24    1
#> 25 40-59 27.4  no 186 25    0
#>
```

```
#> [[4]]
#>      age  bmi hyp chl id miss
#> 1  20-39 24.9  no 187  1    1
#> 2  40-59 22.7  no 187  2    0
#> 3  20-39 27.4  no 187  3    1
#> 4  60-99 24.9 yes 206  4    1
#> 5  20-39 20.4  no 113  5    0
#> 6  60-99 22.5  no 184  6    1
#> 7  20-39 22.5  no 118  7    0
#> 8  20-39 30.1  no 187  8    0
#> 9  40-59 22.0  no 238  9    0
#> 10 40-59 26.3 yes 187 10    1
#> 11 20-39 29.6  no 229 11    1
#> 12 40-59 27.4  no 204 12    1
#> 13 60-99 21.7  no 206 13    0
#> 14 40-59 28.7 yes 204 14    0
#> 15 20-39 29.6  no 186 15    1
#> 16 20-39 35.3  no 184 16    1
#> 17 60-99 27.2 yes 284 17    0
#> 18 40-59 26.3 yes 199 18    0
#> 19 20-39 35.3  no 218 19    0
#> 20 60-99 25.5 yes 199 20    1
#> 21 20-39 27.5  no 187 21    1
#> 22 20-39 33.2  no 229 22    0
#> 23 20-39 27.5  no 131 23    0
#> 24 60-99 24.9  no 284 24    1
#> 25 40-59 27.4  no 186 25    0
#>
#> [[5]]
#>      age  bmi hyp chl id miss
#> 1  20-39 27.2  no 238  1    1
#> 2  40-59 22.7  no 187  2    0
#> 3  20-39 24.9  no 187  3    1
#> 4  60-99 20.4  no 229  4    1
#> 5  20-39 20.4  no 113  5    0
#> 6  60-99 21.7  no 184  6    1
#> 7  20-39 22.5  no 118  7    0
#> 8  20-39 30.1  no 187  8    0
#> 9  40-59 22.0  no 238  9    0
#> 10 40-59 20.4 yes 187 10    1
```

```
#> 11 20-39 25.5  no 118 11    1
#> 12 40-59 21.7  no 187 12    1
#> 13 60-99 21.7  no 206 13    0
#> 14 40-59 28.7 yes 204 14    0
#> 15 20-39 29.6  no 199 15    1
#> 16 20-39 27.5  no 187 16    1
#> 17 60-99 27.2 yes 284 17    0
#> 18 40-59 26.3 yes 199 18    0
#> 19 20-39 35.3  no 218 19    0
#> 20 60-99 25.5 yes 206 20    1
#> 21 20-39 33.2  no 206 21    1
#> 22 20-39 33.2  no 229 22    0
#> 23 20-39 27.5  no 131 23    0
#> 24 60-99 24.9  no 204 24    1
#> 25 40-59 27.4  no 186 25    0
# record the fits from each data
```

Design, subset and fit

For each imputed dataset, a statistical model is fitted. Before fitting, the data is structured to account for survey design features. Only rows without originally missing outcome and exposure values are used for model fitting. The results of the model fitting for each dataset are stored for later analysis.

```r
require(survey)
fit.list <- vector("list", 5)
for (i in 1:m) {
  analytic.i <- data.list[[i]]
  # assigning survey features = 1
  w.design0 <- svydesign(id=~1, weights=~1,
                         data=analytic.i)
  w.design <- subset(w.design0, miss == 0)
  fit <- svyglm(formula, design=w.design, family=binomial)
  fit.list[[i]] <-  fit
}
```

Pooled results

After fitting models to each imputed dataset, the results are combined or "pooled". This pooled result provides a more robust estimate by considering the variability across the imputed datasets.

```r
require(mitools)
pooled.estimates <- MIcombine(fit.list)
pooled.estimates
#> Multiple imputation results:
#>       MIcombine.default(fit.list)
#>                   results         se
#> (Intercept) -1.769918773 2.73723080
#> chl          0.009747869 0.01499608
#> hypyes      18.128613035 1.05775401

# or you can do it this way
betas<-MIextract(fit.list,fun=coef)
vars<-MIextract(fit.list, fun=vcov)
summary(MIcombine(betas,vars))
#> Multiple imputation results:
#>       MIcombine.default(betas, vars)
#>                   results         se      (lower      upper) missInfo
#> (Intercept) -1.769918773 2.73723080 -7.13479255  3.59495501      0 %
#> chl          0.009747869 0.01499608 -0.01964391  0.03913964      0 %
#> hypyes      18.128613035 1.05775401 16.05545326 20.20177281      0 %

# report beta coef
sum.pooled <- summary(pooled.estimates, digits = 2)
#> Multiple imputation results:
#>       MIcombine.default(fit.list)
#>             results    se (lower upper) missInfo
#> (Intercept) -1.7699 2.737  -7.13  3.595      0 %
#> chl          0.0097 0.015  -0.02  0.039      0 %
#> hypyes      18.1286 1.058  16.06 20.202      0 %
sum.pooled
#>                   results         se      (lower      upper) missInfo
#> (Intercept) -1.769918773 2.73723080 -7.13479255  3.59495501      0 %
#> chl          0.009747869 0.01499608 -0.01964391  0.03913964      0 %
#> hypyes      18.128613035 1.05775401 16.05545326 20.20177281      0 %
```

Report OR

The pooled results are further processed to calculate and report odds ratios, which provide insights into the relationships between variables in the context of logistic regression.

```r
sum.pooled.OR <- summary(pooled.estimates, logeffect=TRUE, digits = 2)
#> Multiple imputation results:
#>       MIcombine.default(fit.list)
#>             results      se  (lower  upper) missInfo
#> (Intercept) 1.7e-01 4.7e-01 8.0e-04 3.6e+01      0 %
#> chl         1.0e+00 1.5e-02 9.8e-01 1.0e+00      0 %
#> hypyes      7.5e+07 7.9e+07 9.4e+06 5.9e+08      0 %
sum.pooled.OR
#>                  results           se       (lower       upper) missInfo
#> (Intercept) 1.703468e-01 4.662786e-01 7.968911e-04 3.641406e+01      0 %
#> chl         1.009796e+00 1.514297e-02 9.805478e-01 1.039916e+00      0 %
#> hypyes      7.467180e+07 7.898439e+07 9.392793e+06 5.936336e+08      0 %
```

Using publish package may be possible, but requires complicated process. Look for `publish.MIresult` (but this can be complicated).

Performance with NA

This is a tutorial of how to estimate model performance while analyzing survey data with missing values (for predictive goals).

```
# Load required packages
library(survey)
library(ROCR)
library(WeightedROC)
```

Useful functions

The functions below could be helpful in

- `svyROCw3`: calculating area under the ROC curve (AUC) value with survey data with logistic regression
- `AL.gof3`: testing Archer-Lemeshow goodness of fit for survey data with logistic regression

```
knitr::opts_chunk$set(echo = TRUE)

# ROC curve for Survey Data with Logistic Regression
svyROCw3 <- function(fit=fit7,outcome=analytic2$CVD=="event", weight = NULL,plot=FALSE){
  if (is.null(weight)){ # require(ROCR)
    prob <- predict(fit, type = "response")
  pred <- prediction(as.vector(prob), outcome)
  perf <- performance(pred, "tpr", "fpr")
  auc <- performance(pred, measure = "auc")
  auc <- auc@y.values[[1]]
  if (plot == TRUE){
    roc.data <- data.frame(fpr = unlist(perf@x.values), tpr = unlist(perf@y.values),
      model = "Logistic")
```

```r
    with(data = roc.data,plot(fpr, tpr, type="l", xlim=c(0,1), ylim=c(0,1), lwd=1,
     xlab="1 - specificity", ylab="Sensitivity",
     main = paste("AUC = ", round(auc,3))))
  mtext("Unweighted ROC")
  abline(0,1, lty=2)
  }
  } else { # library(WeightedROC)
    outcome <- as.numeric(outcome)
  pred <- predict(fit, type = "response")
  tp.fp <- WeightedROC(pred, outcome, weight)
  auc <- WeightedAUC(tp.fp)
  if (plot == TRUE){
    with(data = tp.fp,plot(FPR, TPR, type="l", xlim=c(0,1), ylim=c(0,1), lwd=1,
     xlab="1 - specificity", ylab="Sensitivity",
     main = paste("AUC = ", round(auc,3))))
  abline(0,1, lty=2)
  mtext("Weighted ROC")
  }
  }
  return(auc)
}

# Archer-Lemeshow Goodness of Fit Test for Survey Data with Logistic Regression
AL.gof3 <- function(fit=fit7, data = analytic2, weight = "weight", psu = "psu",
                    strata= "strata"){
  r <- residuals(fit, type="response")
  f<-fitted(fit)
  breaks.g <- c(-Inf, quantile(f,  (1:9)/10), Inf)
  breaks.g <- breaks.g + seq_along(breaks.g) * .Machine$double.eps
  g<- cut(f, breaks.g)
  data2g <- cbind(data,r,g)
  if (is.null(psu)){
    newdesign <- svydesign(id=~1,
                           weights=as.formula(paste0("~",weight)),
                          data=data2g, nest = TRUE)
  }
  if (!is.null(psu)) {
    newdesign <- svydesign(id=as.formula(paste0("~",psu)),
                           strata=as.formula(paste0("~",strata)),
                           weights=as.formula(paste0("~",weight)),
```

```r
                        data=data2g, nest = TRUE)
  }
  decilemodel<- svyglm(r~g, design=newdesign)
  res <- regTermTest(decilemodel, ~g)
  return(as.numeric(res$p))
}
```

Load imputed 5 sets of data

Saved at the end of Lab 6 part 2.

```r
# Saved from last lab
load("Data/missingdata/missOA123CVDnorth.RData")
str(allImputations)
#> List of 2
#>  $ imputations:List of 5
#>   ..$ :'data.frame': 135448 obs. of  19 variables:
#>   .. ..$ .imp   : int [1:135448] 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ .id    : int [1:135448] 1 2 3 4 5 6 7 8 9 10 ...
#>   .. ..$ CVD    : Factor w/ 2 levels "0 event","event": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ age    : Factor w/ 4 levels "20-39 years",..: 1 2 1 2 3 1 3 2 1 2 ...
#>   .. ..$ sex    : Factor w/ 2 levels "Female","Male": 2 2 1 2 2 2 1 1 1 1 ...
#>   .. ..$ income : Factor w/ 4 levels "$29,999 or less",..: 4 1 3 4 2 2 1 4 4 3 ...
#>   .. ..$ race   : Factor w/ 2 levels "Non-white","White": 2 2 2 2 2 2 2 2 2 2 ...
#>   .. ..$ bmicat : Factor w/ 3 levels "Normal","Overweight",..: 1 2 1 1 2 1 2 2 3 2 ...
#>   .. ..$ phyact : Factor w/ 3 levels "Active","Inactive",..: 2 1 2 2 3 3 2 1 2 2 ...
#>   .. ..$ smoke  : Factor w/ 3 levels "Current smoker",..: 2 2 2 2 2 3 2 2 3 3 ...
#>   .. ..$ fruit  : Factor w/ 3 levels "0-3 daily serving",..: 2 2 2 2 2 1 2 3 3 2 ...
#>   .. ..$ painmed: Factor w/ 2 levels "No","Yes": 2 1 2 1 1 2 2 2 2 2 ...
#>   .. ..$ ht     : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 2 1 1 1 ...
#>   .. ..$ copd   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ diab   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ edu    : Factor w/ 4 levels "< 2ndary","2nd grad.",..: 4 4 4 4 2 4 1 4 4 4 ...
#>   .. ..$ weight : num [1:135448] 56.1 56.1 39.2 44 59.9 ...
#>   .. ..$ OA     : Factor w/ 2 levels "Control","OA": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ ID     : int [1:135448] 1 3 6 7 12 13 14 16 20 21 ...
#>   ..$ :'data.frame': 135448 obs. of  19 variables:
#>   .. ..$ .imp   : int [1:135448] 2 2 2 2 2 2 2 2 2 2 ...
#>   .. ..$ .id    : int [1:135448] 1 2 3 4 5 6 7 8 9 10 ...
```

```
#>   .. ..$ CVD    : Factor w/ 2 levels "0 event","event": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ age    : Factor w/ 4 levels "20-39 years",..: 1 2 1 2 3 1 3 2 1 2 ...
#>   .. ..$ sex    : Factor w/ 2 levels "Female","Male": 2 2 1 2 2 2 1 1 1 1 ...
#>   .. ..$ income : Factor w/ 4 levels "$29,999 or less",..: 4 1 3 4 2 2 1 4 4 3 ...
#>   .. ..$ race   : Factor w/ 2 levels "Non-white","White": 2 2 2 2 2 2 2 2 2 2 ...
#>   .. ..$ bmicat : Factor w/ 3 levels "Normal","Overweight",..: 1 2 1 1 2 1 2 2 3 2 ...
#>   .. ..$ phyact : Factor w/ 3 levels "Active","Inactive",..: 2 1 2 2 3 3 2 1 2 2 ...
#>   .. ..$ smoke  : Factor w/ 3 levels "Current smoker",..: 2 2 2 2 2 3 2 2 3 3 ...
#>   .. ..$ fruit  : Factor w/ 3 levels "0-3 daily serving",..: 2 2 2 2 2 1 2 3 3 2 ...
#>   .. ..$ painmed: Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 2 2 ...
#>   .. ..$ ht     : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 2 1 1 1 ...
#>   .. ..$ copd   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ diab   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ edu    : Factor w/ 4 levels "< 2ndary","2nd grad.",..: 4 4 4 4 2 4 1 4 4 4 ...
#>   .. ..$ weight : num [1:135448] 56.1 56.1 39.2 44 59.9 ...
#>   .. ..$ OA     : Factor w/ 2 levels "Control","OA": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ ID     : int [1:135448] 1 3 6 7 12 13 14 16 20 21 ...
#>   ..$ :'data.frame': 135448 obs. of  19 variables:
#>   .. ..$ .imp   : int [1:135448] 3 3 3 3 3 3 3 3 3 3 ...
#>   .. ..$ .id    : int [1:135448] 1 2 3 4 5 6 7 8 9 10 ...
#>   .. ..$ CVD    : Factor w/ 2 levels "0 event","event": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ age    : Factor w/ 4 levels "20-39 years",..: 1 2 1 2 3 1 3 2 1 2 ...
#>   .. ..$ sex    : Factor w/ 2 levels "Female","Male": 2 2 1 2 2 2 1 1 1 1 ...
#>   .. ..$ income : Factor w/ 4 levels "$29,999 or less",..: 4 1 3 4 2 2 1 4 4 3 ...
#>   .. ..$ race   : Factor w/ 2 levels "Non-white","White": 2 2 2 2 2 2 2 2 2 2 ...
#>   .. ..$ bmicat : Factor w/ 3 levels "Normal","Overweight",..: 1 2 1 1 2 1 2 2 3 2 ...
#>   .. ..$ phyact : Factor w/ 3 levels "Active","Inactive",..: 2 1 2 2 3 3 2 1 2 2 ...
#>   .. ..$ smoke  : Factor w/ 3 levels "Current smoker",..: 2 2 2 2 2 3 2 2 3 3 ...
#>   .. ..$ fruit  : Factor w/ 3 levels "0-3 daily serving",..: 2 2 2 2 2 1 2 3 3 2 ...
#>   .. ..$ painmed: Factor w/ 2 levels "No","Yes": 2 2 2 1 2 2 2 2 2 2 ...
#>   .. ..$ ht     : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 2 1 1 1 ...
#>   .. ..$ copd   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ diab   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ edu    : Factor w/ 4 levels "< 2ndary","2nd grad.",..: 4 4 4 4 2 4 1 4 4 4 ...
#>   .. ..$ weight : num [1:135448] 56.1 56.1 39.2 44 59.9 ...
#>   .. ..$ OA     : Factor w/ 2 levels "Control","OA": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ ID     : int [1:135448] 1 3 6 7 12 13 14 16 20 21 ...
#>   ..$ :'data.frame': 135448 obs. of  19 variables:
#>   .. ..$ .imp   : int [1:135448] 4 4 4 4 4 4 4 4 4 4 ...
#>   .. ..$ .id    : int [1:135448] 1 2 3 4 5 6 7 8 9 10 ...
```

```
#>   .. ..$ CVD    : Factor w/ 2 levels "0 event","event": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ age    : Factor w/ 4 levels "20-39 years",..: 1 2 1 2 3 1 3 2 1 2 ...
#>   .. ..$ sex    : Factor w/ 2 levels "Female","Male": 2 2 1 2 2 2 1 1 1 1 ...
#>   .. ..$ income : Factor w/ 4 levels "$29,999 or less",..: 4 1 3 4 2 2 1 4 4 3 ...
#>   .. ..$ race   : Factor w/ 2 levels "Non-white","White": 2 2 2 2 2 2 2 2 2 2 ...
#>   .. ..$ bmicat : Factor w/ 3 levels "Normal","Overweight",..: 1 2 1 1 2 1 2 2 3 2 ...
#>   .. ..$ phyact : Factor w/ 3 levels "Active","Inactive",..: 2 1 2 2 3 3 2 1 2 2 ...
#>   .. ..$ smoke  : Factor w/ 3 levels "Current smoker",..: 2 2 2 2 2 3 2 2 3 3 ...
#>   .. ..$ fruit  : Factor w/ 3 levels "0-3 daily serving",..: 2 2 2 2 2 1 2 3 3 2 ...
#>   .. ..$ painmed: Factor w/ 2 levels "No","Yes": 2 2 2 2 1 1 2 1 2 2 ...
#>   .. ..$ ht     : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 2 1 1 1 ...
#>   .. ..$ copd   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ diab   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ edu    : Factor w/ 4 levels "< 2ndary","2nd grad.",..: 4 4 4 4 2 4 1 4 4 4 ...
#>   .. ..$ weight : num [1:135448] 56.1 56.1 39.2 44 59.9 ...
#>   .. ..$ OA     : Factor w/ 2 levels "Control","OA": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ ID     : int [1:135448] 1 3 6 7 12 13 14 16 20 21 ...
#>   ..$ :'data.frame': 135448 obs. of  19 variables:
#>   .. ..$ .imp   : int [1:135448] 5 5 5 5 5 5 5 5 5 5 ...
#>   .. ..$ .id    : int [1:135448] 1 2 3 4 5 6 7 8 9 10 ...
#>   .. ..$ CVD    : Factor w/ 2 levels "0 event","event": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ age    : Factor w/ 4 levels "20-39 years",..: 1 2 1 2 3 1 3 2 1 2 ...
#>   .. ..$ sex    : Factor w/ 2 levels "Female","Male": 2 2 1 2 2 2 1 1 1 1 ...
#>   .. ..$ income : Factor w/ 4 levels "$29,999 or less",..: 4 1 3 4 2 2 1 4 4 3 ...
#>   .. ..$ race   : Factor w/ 2 levels "Non-white","White": 2 2 2 2 2 2 2 2 2 2 ...
#>   .. ..$ bmicat : Factor w/ 3 levels "Normal","Overweight",..: 1 2 1 1 2 1 2 2 3 2 ...
#>   .. ..$ phyact : Factor w/ 3 levels "Active","Inactive",..: 2 1 2 2 3 3 2 1 2 2 ...
#>   .. ..$ smoke  : Factor w/ 3 levels "Current smoker",..: 2 2 2 2 2 3 2 2 3 3 ...
#>   .. ..$ fruit  : Factor w/ 3 levels "0-3 daily serving",..: 2 2 2 2 2 1 2 3 3 2 ...
#>   .. ..$ painmed: Factor w/ 2 levels "No","Yes": 1 1 2 2 2 2 1 2 2 2 ...
#>   .. ..$ ht     : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 2 1 1 1 ...
#>   .. ..$ copd   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ diab   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ edu    : Factor w/ 4 levels "< 2ndary","2nd grad.",..: 4 4 4 4 2 4 1 4 4 4 ...
#>   .. ..$ weight : num [1:135448] 56.1 56.1 39.2 44 59.9 ...
#>   .. ..$ OA     : Factor w/ 2 levels "Control","OA": 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ ID     : int [1:135448] 1 3 6 7 12 13 14 16 20 21 ...
#>  $ call       : language imputationList(list(subset(impdata, subset = .imp == 1), subset(im
#>  - attr(*, "class")= chr "imputationList"
```

Estimating treatment effect

Individual beta estimates

```r
library(survey)
w.design <- svydesign(ids=~1, weights=~weight,
                           data = allImputations)
model.formula <- as.formula("I(CVD=='event') ~ OA + age + sex +
                            income + race + painmed + ht +
                            copd + diab + OA:painmed +
                            age:copd + sex:copd")
estimates <- with(w.design, svyglm(model.formula, family=quasibinomial))
estimates
#> [[1]]
#> Independent Sampling design (with replacement)
#> svydesign(ids = ids, probs = probs, strata = strata, variables = variables,
#>     fpc = fpc, nest = nest, check.strata = check.strata, weights = weights,
#>     data = d, pps = pps, ...)
#>
#> Call:  svyglm(formula = model.formula, design = .design, family = quasibinomial)
#>
#> Coefficients:
#>            (Intercept)                    OAOA          age40-49 years
#>                -5.6809                  1.1063                  0.7911
#>         age50-59 years          age60-64 years                 sexMale
#>                 1.6233                  2.0076                  0.6278
#>  income$30,000-$49,999   income$50,000-$79,999   income$80,000 or more
#>                -0.4347                 -0.6031                 -0.6805
#>              raceWhite              painmedYes                   htYes
#>                 0.2136                  0.8277                  1.0344
#>                copdYes                 diabYes         OAOA:painmedYes
#>                 1.9143                  0.8166                 -0.8183
#> age40-49 years:copdYes  age50-59 years:copdYes  age60-64 years:copdYes
#>                -1.1404                 -0.7286                 -0.9360
#>        sexMale:copdYes
#>                 0.6167
#>
#> Degrees of Freedom: 135447 Total (i.e. Null);  135429 Residual
#> Null Deviance:       36680
#> Residual Deviance: 31100     AIC: NA
```

```
#>
#> [[2]]
#> Independent Sampling design (with replacement)
#> svydesign(ids = ids, probs = probs, strata = strata, variables = variables,
#>     fpc = fpc, nest = nest, check.strata = check.strata, weights = weights,
#>     data = d, pps = pps, ...)
#>
#> Call:  svyglm(formula = model.formula, design = .design, family = quasibinomial)
#>
#> Coefficients:
#>            (Intercept)                    OAOA          age40-49 years
#>                -5.4837                  0.8766                  0.7781
#>         age50-59 years          age60-64 years                  sexMale
#>                 1.6090                  1.9901                  0.6083
#>  income$30,000-$49,999   income$50,000-$79,999   income$80,000 or more
#>                -0.4415                 -0.5969                 -0.6840
#>              raceWhite              painmedYes                   htYes
#>                 0.2374                  0.5877                  1.0413
#>                copdYes                 diabYes         OAOA:painmedYes
#>                 1.8837                  0.8054                 -0.5388
#> age40-49 years:copdYes  age50-59 years:copdYes  age60-64 years:copdYes
#>                -1.1303                 -0.6944                 -0.8545
#>        sexMale:copdYes
#>                 0.5857
#>
#> Degrees of Freedom: 135447 Total (i.e. Null);  135429 Residual
#> Null Deviance:       36680
#> Residual Deviance: 31260     AIC: NA
#>
#> [[3]]
#> Independent Sampling design (with replacement)
#> svydesign(ids = ids, probs = probs, strata = strata, variables = variables,
#>     fpc = fpc, nest = nest, check.strata = check.strata, weights = weights,
#>     data = d, pps = pps, ...)
#>
#> Call:  svyglm(formula = model.formula, design = .design, family = quasibinomial)
#>
#> Coefficients:
#>            (Intercept)                    OAOA          age40-49 years
#>                -5.6365                  1.0389                  0.7815
```

```
#>         age50-59 years          age60-64 years                 sexMale
#>                 1.6140                  1.9986                  0.6180
#>  income$30,000-$49,999   income$50,000-$79,999   income$80,000 or more
#>                -0.4334                 -0.5945                 -0.6843
#>              raceWhite              painmedYes                   htYes
#>                 0.2042                  0.7963                  1.0386
#>                copdYes                 diabYes         OAOA:painmedYes
#>                 1.9468                  0.8020                 -0.7330
#> age40-49 years:copdYes  age50-59 years:copdYes  age60-64 years:copdYes
#>                -1.1185                 -0.7775                 -0.9414
#>        sexMale:copdYes
#>                 0.5548
#>
#> Degrees of Freedom: 135447 Total (i.e. Null);  135429 Residual
#> Null Deviance:       36680
#> Residual Deviance: 31130     AIC: NA
#>
#> [[4]]
#> Independent Sampling design (with replacement)
#> svydesign(ids = ids, probs = probs, strata = strata, variables = variables,
#>     fpc = fpc, nest = nest, check.strata = check.strata, weights = weights,
#>     data = d, pps = pps, ...)
#>
#> Call:  svyglm(formula = model.formula, design = .design, family = quasibinomial)
#>
#> Coefficients:
#>            (Intercept)                    OAOA          age40-49 years
#>                -5.6811                  1.2937                  0.7838
#>         age50-59 years          age60-64 years                 sexMale
#>                 1.6248                  2.0097                  0.6285
#>  income$30,000-$49,999   income$50,000-$79,999   income$80,000 or more
#>                -0.4431                 -0.6012                 -0.6877
#>              raceWhite              painmedYes                   htYes
#>                 0.2325                  0.8179                  1.0401
#>                copdYes                 diabYes         OAOA:painmedYes
#>                 1.9368                  0.8022                 -1.0624
#> age40-49 years:copdYes  age50-59 years:copdYes  age60-64 years:copdYes
#>                -1.0381                 -0.7407                 -0.9356
#>        sexMale:copdYes
#>                 0.5422
```

```
#>
#> Degrees of Freedom: 135447 Total (i.e. Null);  135429 Residual
#> Null Deviance:       36680
#> Residual Deviance: 31100     AIC: NA
#>
#> [[5]]
#> Independent Sampling design (with replacement)
#> svydesign(ids = ids, probs = probs, strata = strata, variables = variables,
#>     fpc = fpc, nest = nest, check.strata = check.strata, weights = weights,
#>     data = d, pps = pps, ...)
#>
#> Call:  svyglm(formula = model.formula, design = .design, family = quasibinomial)
#>
#> Coefficients:
#>            (Intercept)                    OAOA          age40-49 years
#>                -5.4487                  0.7569                  0.7752
#>         age50-59 years          age60-64 years                  sexMale
#>                 1.6021                  1.9847                  0.6107
#>  income$30,000-$49,999   income$50,000-$79,999   income$80,000 or more
#>                -0.4393                 -0.5977                 -0.6788
#>              raceWhite              painmedYes                   htYes
#>                 0.2351                  0.5457                  1.0421
#>                copdYes                 diabYes         OAOA:painmedYes
#>                 1.9235                  0.8057                 -0.3884
#> age40-49 years:copdYes  age50-59 years:copdYes  age60-64 years:copdYes
#>                -1.1275                 -0.7370                 -0.9152
#>        sexMale:copdYes
#>                 0.5871
#>
#> Degrees of Freedom: 135447 Total (i.e. Null);  135429 Residual
#> Null Deviance:       36680
#> Residual Deviance: 31290     AIC: NA
#>
#> attr(,"call")
#> with(w.design, svyglm(model.formula, family = quasibinomial))
```

Pooled / averaged estimates for beta and OR

```r
library("mitools")
pooled.estimates <- MIcombine(estimates)
pooled.estimates
#> Multiple imputation results:
#>       with(w.design, svyglm(model.formula, family = quasibinomial))
#>       MIcombine.default(estimates)
#>                            results         se
#> (Intercept)            -5.5861842 0.19239878
#> OAOA                    1.0144758 0.27097789
#> age40-49 years          0.7819429 0.10165218
#> age50-59 years          1.6146502 0.09905167
#> age60-64 years          1.9981358 0.10380099
#> sexMale                 0.6186653 0.05484262
#> income$30,000-$49,999  -0.4383959 0.06787453
#> income$50,000-$79,999  -0.5987066 0.06593903
#> income$80,000 or more  -0.6830429 0.06907307
#> raceWhite               0.2245784 0.10567063
#> painmedYes              0.7150386 0.16508180
#> htYes                   1.0393147 0.05522682
#> copdYes                 1.9210219 0.65240500
#> diabYes                 0.8063590 0.08169615
#> OAOA:painmedYes        -0.7081707 0.32575000
#> age40-49 years:copdYes -1.1109564 0.69441469
#> age50-59 years:copdYes -0.7356335 0.67352566
#> age60-64 years:copdYes -0.9165474 0.65626790
#> sexMale:copdYes         0.5772949 0.30888502

sum.pooled <- summary(pooled.estimates,logeffect=TRUE, digits = 2)
#> Multiple imputation results:
#>       with(w.design, svyglm(model.formula, family = quasibinomial))
#>       MIcombine.default(estimates)
#>                        results      se (lower  upper) missInfo
#> (Intercept)             0.0037 0.00072 0.0025  0.0056     45 %
#> OAOA                    2.7579 0.74733 1.4778  5.1469     76 %
#> age40-49 years          2.1857 0.22218 1.7909  2.6676      0 %
#> age50-59 years          5.0261 0.49785 4.1392  6.1031      1 %
#> age60-64 years          7.3753 0.76556 6.0175  9.0394      1 %
#> sexMale                 1.8564 0.10181 1.6672  2.0672      4 %
```

```
#> income$30,000-$49,999   0.6451 0.04378 0.5647  0.7369      0 %
#> income$50,000-$79,999   0.5495 0.03623 0.4829  0.6253      0 %
#> income$80,000 or more   0.5051 0.03489 0.4411  0.5783      0 %
#> raceWhite               1.2518 0.13228 1.0176  1.5399      2 %
#> painmedYes              2.0443 0.33747 1.3628  3.0664     86 %
#> htYes                   2.8273 0.15614 2.5372  3.1505      0 %
#> copdYes                 6.8279 4.45458 1.9009 24.5255      0 %
#> diabYes                 2.2397 0.18298 1.9083  2.6287      1 %
#> OAOA:painmedYes         0.4925 0.16045 0.2275  1.0663     81 %
#> age40-49 years:copdYes  0.3292 0.22863 0.0844  1.2841      0 %
#> age50-59 years:copdYes  0.4792 0.32275 0.1280  1.7940      0 %
#> age60-64 years:copdYes  0.3999 0.26244 0.1105  1.4473      0 %
#> sexMale:copdYes         1.7812 0.55019 0.9723  3.2632      1 %
sum.pooled
#>                           results           se      (lower       upper)
#> (Intercept)            0.003749307 0.0007213621 0.002521474  0.005575035
#> OAOA                   2.757917293 0.7473346089 1.477812451  5.146869475
#> age40-49 years         2.185714763 0.2221826634 1.790880131  2.667598431
#> age50-59 years         5.026129638 0.4978465465 4.139210342  6.103091424
#> age60-64 years         7.375294204 0.7655628729 6.017543434  9.039397089
#> sexMale                1.856448612 0.1018124982 1.667183736  2.067199538
#> income$30,000-$49,999  0.645070326 0.0437838454 0.564718199  0.736855526
#> income$50,000-$79,999  0.549521954 0.0362349439 0.482900282  0.625334856
#> income$80,000 or more  0.505077756 0.0348872707 0.441126323  0.578300423
#> raceWhite              1.251794895 0.1322779501 1.017586096  1.539909463
#> painmedYes             2.044265556 0.3374710393 1.362836413  3.066414738
#> htYes                  2.827278767 0.1561416172 2.537226797  3.150489043
#> copdYes                6.827932558 4.4545773482 1.900905879 24.525497836
#> diabYes                2.239738282 0.1829779905 1.908343389  2.628681819
#> OAOA:painmedYes        0.492544363 0.1604463252 0.227516338  1.066296826
#> age40-49 years:copdYes 0.329243929 0.2286318203 0.084416559  1.284126781
#> age50-59 years:copdYes 0.479201771 0.3227546865 0.128000420  1.794012369
#> age60-64 years:copdYes 0.399897335 0.2624397832 0.110491796  1.447328067
#> sexMale:copdYes        1.781213579 0.5501901952 0.972261649  3.263238672
#>                        missInfo
#> (Intercept)                45 %
#> OAOA                       76 %
#> age40-49 years              0 %
#> age50-59 years              1 %
#> age60-64 years              1 %
```

```
#> sexMale                        4 %
#> income$30,000-$49,999          0 %
#> income$50,000-$79,999          0 %
#> income$80,000 or more          0 %
#> raceWhite                      2 %
#> painmedYes                    86 %
#> htYes                          0 %
#> copdYes                        0 %
#> diabYes                        1 %
#> OAOA:painmedYes               81 %
#> age40-49 years:copdYes         0 %
#> age50-59 years:copdYes         0 %
#> age60-64 years:copdYes         0 %
#> sexMale:copdYes                1 %
```

Estimating model performance (AUC and AL)

Individual AUC estimates (with interactions)

```r
library(ROCR)
library(WeightedROC)
model.formula <- as.formula("I(CVD=='event') ~ OA + age + sex +
                             income + race + painmed + ht +
                             copd + diab + OA:painmed +
                             age:copd + sex:copd")
AL.scalar <- AUC.scalar <- vector("list", 5)
for (i in 1:5) {
  analytic.i <- allImputations$imputations[[i]]
  w.design <- svydesign(id=~1, weights=~weight,
                        data=analytic.i)
  model.fit <- svyglm(model.formula, design=w.design, family=quasibinomial)
  auc <- svyROCw3(fit=model.fit,outcome=w.design$variables$CVD=='event',
                  weight = w.design$variables$weight, plot = FALSE)

  AL <- AL.gof3(fit=model.fit, data = analytic.i,
                   weight = "weight",
                   psu = NULL,
                   strata= NULL)
```

```r
  AL.scalar[[i]] <- AL
  AUC.scalar[[i]] <- auc
  cat("AUC calculated for data", i, "\n")
}
#> AUC calculated for data 1
#> AUC calculated for data 2
#> AUC calculated for data 3
#> AUC calculated for data 4
#> AUC calculated for data 5
str(AUC.scalar)
#> List of 5
#>  $ : num 0.8
#>  $ : num 0.795
#>  $ : num 0.798
#>  $ : num 0.8
#>  $ : num 0.794
AL.scalar
#> [[1]]
#> [1] 0.01243738
#>
#> [[2]]
#> [1] 0.5471927
#>
#> [[3]]
#> [1] 0.13081
#>
#> [[4]]
#> [1] 0.644286
#>
#> [[5]]
#> [1] 0.6049137
```

Averaged estimates for AUC (with interactions)

```r
# summary of AUC
mean(unlist(AUC.scalar))
#> [1] 0.7973746
sd(unlist(AUC.scalar))
```

```r
#> [1] 0.002688964
round(range(unlist(AUC.scalar)),3)
#> [1] 0.794 0.800
# p-values (from AL) by majority
sum(AL.scalar>0.05)
#> [1] 4
```

Model performance without interactions

Individual AUC estimates / AL p-values

```r
library(ROCR)
library(WeightedROC)
AL.scalar <- AUC.scalar <- vector("list", 5)
model.formula <- as.formula("I(CVD=='event') ~ OA + age + sex +
                            income + race + painmed + ht +
                            copd + diab")
for (i in 1:5) {
  analytic.i <- allImputations$imputations[[i]]
  w.design <- svydesign(id=~1, weights=~weight,
                        data=analytic.i)
  model.fit <- svyglm(model.formula, design=w.design, family=quasibinomial)
  auc <- svyROCw3(fit=model.fit,outcome=w.design$variables$CVD=='event',
                  weight = w.design$variables$weight, plot = FALSE)

  AL <- AL.gof3(fit=model.fit, data = analytic.i,
                   weight = "weight",
                   psu = NULL,
                   strata= NULL)
  AL.scalar[[i]] <- AL
  AUC.scalar[[i]] <- auc
  cat("AUC calculated for data", i, "\n")
}
#> AUC calculated for data 1
#> AUC calculated for data 2
#> AUC calculated for data 3
#> AUC calculated for data 4
#> AUC calculated for data 5
```

```r
str(AUC.scalar)
#> List of 5
#>  $ : num 0.798
#>  $ : num 0.794
#>  $ : num 0.797
#>  $ : num 0.798
#>  $ : num 0.794
AL.scalar
#> [[1]]
#> [1] 0.01839624
#>
#> [[2]]
#> [1] 0.2836256
#>
#> [[3]]
#> [1] 0.02439659
#>
#> [[4]]
#> [1] 0.8333214
#>
#> [[5]]
#> [1] 0.2050434
```

Averaged estimates for AUC / majority of AL p-values

```r
# summary of AUC
mean(unlist(AUC.scalar))
#> [1] 0.7964061
sd(unlist(AUC.scalar))
#> [1] 0.002002636
round(range(unlist(AUC.scalar)),3)
#> [1] 0.794 0.798
# p-values (from AL) by majority
sum(AL.scalar>0.05)
#> [1] 3
```

Appendix

User-written `svyROCw3` and `AL.gof3` functions

```
svyROCw3
#> function(fit=fit7,outcome=analytic2$CVD=="event", weight = NULL,plot=FALSE){
#>   # ROC curve for
#>   # Survey Data with Logistic Regression
#>   if (is.null(weight)){ # require(ROCR)
#>     prob <- predict(fit, type = "response")
#>   pred <- prediction(as.vector(prob), outcome)
#>   perf <- performance(pred, "tpr", "fpr")
#>   auc <- performance(pred, measure = "auc")
#>   auc <- auc@y.values[[1]]
#>   if (plot == TRUE){
#>     roc.data <- data.frame(fpr = unlist(perf@x.values), tpr = unlist(perf@y.values),
#>       model = "Logistic")
#>     with(data = roc.data,plot(fpr, tpr, type="l", xlim=c(0,1), ylim=c(0,1), lwd=1,
#>      xlab="1 - specificity", ylab="Sensitivity",
#>      main = paste("AUC = ", round(auc,3))))
#>   mtext("Unweighted ROC")
#>   abline(0,1, lty=2)
#>   }
#>   } else { # library(WeightedROC)
#>     outcome <- as.numeric(outcome)
#>   pred <- predict(fit, type = "response")
#>   tp.fp <- WeightedROC(pred, outcome, weight)
#>   auc <- WeightedAUC(tp.fp)
#>   if (plot == TRUE){
#>     with(data = tp.fp,plot(FPR, TPR, type="l", xlim=c(0,1), ylim=c(0,1), lwd=1,
#>      xlab="1 - specificity", ylab="Sensitivity",
#>      main = paste("AUC = ", round(auc,3))))
#>   abline(0,1, lty=2)
#>   mtext("Weighted ROC")
#>   }
#>   }
#>   return(auc)
#> }
#> <bytecode: 0x000002235cbbdfd8>
AL.gof3
#> function(fit=fit7, data = analytic2, weight = "weight", psu = "psu", strata= "strata"){
```

```
#>   # Archer-Lemeshow Goodness of Fit Test for
#>   # Survey Data with Logistic Regression
#>   r <- residuals(fit, type="response")
#>   f<-fitted(fit)
#>   breaks.g <- c(-Inf, quantile(f,  (1:9)/10), Inf)
#>   breaks.g <- breaks.g + seq_along(breaks.g) * .Machine$double.eps
#>   g<- cut(f, breaks.g)
#>   data2g <- cbind(data,r,g)
#>   if (is.null(psu)){
#>     newdesign <- svydesign(id=~1,
#>                           weights=as.formula(paste0("~",weight)),
#>                          data=data2g, nest = TRUE)
#>   }
#>   if (!is.null(psu)) {
#>     newdesign <- svydesign(id=as.formula(paste0("~",psu)),
#>                           strata=as.formula(paste0("~",strata)),
#>                           weights=as.formula(paste0("~",weight)),
#>                          data=data2g, nest = TRUE)
#>   }
#>   decilemodel<- svyglm(r~g, design=newdesign)
#>   res <- regTermTest(decilemodel, ~g)
#>   return(as.numeric(res$p))
#> }
#> <bytecode: 0x000002235d1f0ea8>
```

Subpopulations

This tutorial demonstrates how to manage missing data in complex surveys using multiple imputation, focusing on specific subpopulations defined by the study's eligibility criteria.

Purpose

Let us we are interested in exploring the relationship between **rheumatoid arthritis** and **cardiovascular disease** (CVD) among US adults aged 20 years or more. For that, we will use NHANES 2017–2018 dataset, which follows a complex survey design.

The purpose of this example is to *demonstrate how to do the missing data analysis with multiple imputation in the context of complex surveys.*

In this tutorial, we used a similar approach to the one in a published article by Hossain et al. (2022), but we used less data (restricted to only 2017–2018) to speed up the analysis. Ref link.

The main idea is:

- working with the analytic data
- imputing missing values based on that analytic dataset
- keep count of the ineligible subjects from the full data who are not included in the analytic data
- adding those ineligible subjects back in the imputed datasets, so that we can utilize the survey features and subset the design.

```r
# Load required packages
library(dplyr)
library(kableExtra)
library(tableone)
library(survey)
library(Publish)
library(DataExplorer)
```

```r
library(mice)
library(mitools)
```

Let us import the dataset:

```r
load("Data/missingdata/MIexample.RData")
ls()
#> [1] "dat.full"        "has_annotations"
```

```r
dim(dat.full)
#> [1] 9254   15
head(dat.full)
#>   studyid survey.weight psu strata  cvd rheumatoid age    sex
#> 1   93703      9246.492   2    145 <NA>       <NA>   2 Female
#> 2   93704     37338.768   1    143 <NA>       <NA>   2   Male
#> 3   93705      8614.571   2    145   No        Yes  66 Female
#> 4   93706      8548.633   2    134 <NA>       <NA>  18   Male
#> 5   93707      6769.345   1    138 <NA>       <NA>  13   Male
#> 6   93708     13329.451   2    138   No       <NA>  66 Female
#>               education   race             income  bmi         smoking  htn
#> 1                  <NA> Others   $75,000 and Over 17.5            <NA> <NA>
#> 2                  <NA>  White   $75,000 and Over 15.7            <NA> <NA>
#> 3 Less than high school  Black  less than $20,000 31.7 Previous smoker  Yes
#> 4                  <NA> Others               <NA> 21.5    Never smoker   No
#> 5                  <NA> Others $20,000 to $74,999 18.1            <NA>  Yes
#> 6 Less than high school Others $20,000 to $74,999 23.7    Never smoker  Yes
#>   diabetes
#> 1       No
#> 2       No
#> 3       No
#> 4       No
#> 5       No
#> 6       No
```

The dataset (`dat.full`) contains 9,254 subjects with 15 variables:

Survey information

- `studyid`: Respondent sequence number

- `survey.weight`: Full sample 2 year interview weight
- `psu`: Masked pseudo PSU
- `strata`: Masked pseudo strata

Outcome variable

- `cvd`: Whether having cardiovascular disease

Exposure variable

- `rheumatoid`: Whether having rheumatoid arthritis

Covariates

- `age`: age in years at screening
- `sex`
- `education`
- `race`: Race/Ethnicity
- `income`: Family income in $
- `bmi`: Body Mass Index in kg/m^2
- `smoking`: Smoking status
- `htn`: Having hypertension
- `diabetes`: Having diabetes

Analytic dataset

Subsetting according to eligibility

Let us create an analytic dataset for

- adults aged 20 years or more
- without missing values in outcome (cvd) or exposure (rheumatoid arthritis).

```
# Drop < 20 years
dat.with.miss <- subset(dat.full, age >= 20)

# Frequency for outcome and exposure
table(dat.with.miss$cvd, useNA = "always") # 6 missing
#>
#>   No  Yes <NA>
#> 4872  691    6
```

```r
table(dat.with.miss$rheumatoid, useNA = "always") # 1375 missing
#>
#>   No  Yes <NA>
#> 3857  337 1375

# Drop missing in outcome and exposure
# i.e., dataset with missing values only in covariates
dat.analytic <- dat.with.miss[complete.cases(dat.with.miss$cvd),]
dat.analytic <- dat.analytic[complete.cases(dat.analytic$rheumatoid),]
nrow(dat.analytic)
#> [1] 4191
```

As we can see, we have 4,191 participants aged 20 years or more without missing values in outcome or exposure. Let us count the ineligible subjects from the full data and create an indicator variable.

```r
dat.full$ineligible <- 1
dat.full$ineligible[dat.full$studyid %in% dat.analytic$studyid] <- 0
table(dat.full$ineligible, useNA = "always")
#>
#>    0    1 <NA>
#> 4191 5063    0
```

We have 4,191 eligible and 5,063 ineligible subjects based on the eligibility criteria.

General strategy of solution:

- We will build the imputation model on 4,191 eligible subjects, and
- later we will include 5,063 ineligible subjects in the data so that we can utilize survey features.

Table 1

Let us see the summary statistics, i.e., create Table 1 stratified by outcome (cvd). Before that, we will categorize `age` and recode `rheumatoid`:

```r
# Categorical age
dat.analytic$age.cat <-
  with(dat.analytic, ifelse(age >= 20 & age < 50, "20-49",
                            ifelse(age >= 50 & age < 65,
                                   "50-64", "65+")))
dat.analytic$age.cat <- factor(dat.analytic$age.cat,
                               levels = c("20-49", "50-64", "65+"))
table(dat.analytic$age.cat, useNA = "always")
#>
#> 20-49 50-64   65+  <NA>
#>  2280  1097   814     0

# Recode rheumatoid to arthritis
dat.analytic$arthritis <-
car::recode(dat.analytic$rheumatoid, " 'No' = 'No arthritis';
            'Yes' = 'Rheumatoid arthritis' ", as.factor = T)
table(dat.analytic$arthritis, useNA = "always")
#>
#>         No arthritis Rheumatoid arthritis                 <NA>
#>                 3854                  337                    0

# Keep only relevant variables
vars <-  c("studyid", "survey.weight", "psu", "strata", "cvd",
           "arthritis", "age.cat", "sex", "education", "race",
           "income", "bmi", "smoking", "htn", "diabetes")
dat.analytic2 <- dat.analytic[, vars]
```

```r
# Create Table 1
vars <- c("arthritis", "age.cat", "sex", "education", "race", "income", "bmi", "smoking",
          "htn", "diabetes")
tab1 <- CreateTableOne(vars = vars, strata = "cvd",
                       data = dat.analytic2, includeNA = F,
                       addOverall = T, test = F)
print(tab1, format = "f", showAllLevels = T)
#>                  Stratified by cvd
#>                   level                     Overall      No
#>   n                                          4191         3823
#>   arthritis       No arthritis               3854         3580
#>                   Rheumatoid arthritis        337          243
#>   age.cat         20-49                      2280         2240
```

```
#>                    50-64                       1097          979
#>                    65+                          814          604
#>   sex              Male                        2126         1884
#>                    Female                      2065         1939
#>   education        Less than high school        828          728
#>                    High school                 2292         2094
#>                    College graduate or above   1063          993
#>   race             White                       1275         1113
#>                    Black                        998          898
#>                    Hispanic                    1015          958
#>                    Others                       903          854
#>   income           less than $20,000            659          557
#>                    $20,000 to $74,999          1967         1796
#>                    $75,000 and Over            1143         1079
#>   bmi (mean (SD))                            29.28 (7.19) 29.20 (7.18)
#>   smoking          Never smoker                2570         2427
#>                    Previous smoker              882          726
#>                    Current smoker               739          670
#>   htn              No                          1424         1380
#>                    Yes                         2415         2107
#>   diabetes         No                          3622         3396
#>                    Yes                          566          424
#>                   Stratified by cvd
#>                    Yes
#>   n                  368
#>   arthritis          274
#>                       94
#>   age.cat             40
#>                      118
#>                      210
#>   sex                242
#>                      126
#>   education          100
#>                      198
#>                       70
#>   race               162
#>                      100
#>                       57
#>                       49
#>   income             102
```

```
#>                        171
#>                         64
#>   bmi (mean (SD)) 30.09 (7.29)
#>   smoking              143
#>                        156
#>                         69
#>   htn                   44
#>                        308
#>   diabetes             226
#>                        142
```

Check missingness using a plot

Now we will see the percentage of missing values in the variables.

```
DataExplorer::plot_missing(dat.analytic2)
```

We have about 10% missing values in income, followed by hypertension (8.4%), bmi (6.8%), education (0.2%), and diabetes (0.1%).

Dealing with missing values in covariates

- Now we will perform multiple imputation to deal with missing values only in covariates. We will use the

dat.analytic2 dataset that contains missing values in the covariates but no missing values in the outcome or exposure.

- For this exercise, we will consider 5 imputed datasets, 3 iterations, and the predictive mean matching method for bmi and income.
  - We have already set up the data such that the variables are of appropriate types, e.g., numeric bmi, factor age, sex, and so on.
  - We will use the strata variable as an auxiliary variable in the imputation model but not the survey weight or PSU variable.
  - Now we will set up the initial model and set up the methods and predictor matrix before imputing 5 datasets.

Step 0: Set up the imputation model

```r
# Step 0: Set imputation model
ini <- mice(data = dat.analytic2, maxit = 0, print = FALSE)
pred <- ini$pred

# Use the strata variable as an auxiliary variable in the imputation model
pred["strata",] <- 0

# Do not use survey weight or PSU variable as auxiliary variables
pred[,"studyid"] <- pred["studyid",] <- 0
pred[,"psu"] <- pred["psu",] <- 0
pred[,"survey.weight"] <- pred["survey.weight",] <- 0

# Set imputation method
meth <- ini$meth
meth["bmi"] <- "pmm"
meth["income"] <- "pmm"
meth
#>       studyid survey.weight           psu        strata           cvd
#>            ""            ""            ""            ""            ""
#>     arthritis       age.cat           sex     education          race
```

```
#>            ""            ""            ""     "polyreg"            ""
#>        income           bmi       smoking           htn      diabetes
#>         "pmm"         "pmm"            ""      "logreg"      "logreg"
```

- There is no missing for studyid, survey.weight, psu, strata, cvd, arthritis, age, sex, race, smoking. Hence, no method is assigned for these variables.
- For education, `polyreg` (Polytomous logistic regression) will be used.
- Similarly, we will use `pmm` (Predictive mean matching) for bmi, income and used `logreg` (Logistic regression) for htn, diabetes. See the Imputation chapter to see how the PMM method works.

Step 1: Imputing missing values using mice

0.0.0.0.1 * 1.1 Imputing dataset for eligible subjects

```r
# Step 1: impute the incomplete data
imputation <- mice(data = dat.analytic2,
                   seed = 123,
                   predictorMatrix = pred,
                   method = meth,
                   m = 5,
                   maxit = 3,
                   print = FALSE)
```

Now we will combine m = 5 datasets and create a stacked dataset. This dataset should contain 5*4,191 = 20,955 rows.

```r
impdata <- mice::complete(imputation, action="long")

table(impdata$age.cat)
#>
#> 20-49 50-64   65+
#> 11400  5485  4070
```

Note that age has no missing values, and everyone is above 20.

```r
#Remove .id variable from the model as it was created in an intermediate step
impdata$.id <- NULL

# Create an indicator of eligible subjects
impdata$ineligible <- 0

# Number of subjects
nrow(impdata)
#> [1] 20955
```

Let's see whether there is any missing value after imputation:

```r
DataExplorer::plot_missing(impdata)
```

- There is no missing value after imputation.
- As we can see, there is an additional variable (`.imp`) in the imputed dataset. This `.imp` goes from 1 to m = 5, indicating the first to the fifth imputed datasets.

0.0.0.0.2 * 1.2 Preparing dataset for ineligible subjects

The next task is adding the **ineligible subjects** in the imputed datasets, so that we can set up the survey design on the full dataset and then subset the design.

```r
# Number of ineligible subjects
#dat.full$ineligible <- 1
#dat.full$ineligible[dat.full$studyid %in% dat.analytic$studyid] <- 0
table(dat.full$ineligible, useNA = "always")
#>
#>    0    1 <NA>
#> 4191 5063    0
```

Now we will subset the data for ineligible subjects and create `m = 5` copies.

```r
# Subset for ineligible
dat.ineligible <- subset(dat.full, ineligible == 1)

# Create m = 5 datasets with .imp 1 to m = 5
dat31 <- dat.ineligible; dat31$.imp <- 1
dat32 <- dat.ineligible; dat32$.imp <- 2
dat33 <- dat.ineligible; dat33$.imp <- 3
dat34 <- dat.ineligible; dat34$.imp <- 4
dat35 <- dat.ineligible; dat35$.imp <- 5
```

The next step is combining ineligible datasets. Before merging the `stacked dataset for ineligible subjects` to the `imputed stacked dataset for eligible subjects`, we must ensure the variable names are the same.

```r
# Stacked data for ineligible subjects
dat.ineligible.stacked <- rbind(dat31, dat32, dat33, dat34, dat35)
```

We should have missing value in this ineligible part of the data:

```r
DataExplorer::plot_missing(dat.ineligible.stacked)
```

0.0.0.0.3 * 1.3 Combining eligible (imputed) and ineligible (unimputed) subjects

```
names(impdata)
#>  [1] ".imp"          "studyid"       "survey.weight" "psu"
#>  [5] "strata"        "cvd"           "arthritis"     "age.cat"
#>  [9] "sex"           "education"     "race"          "income"
#> [13] "bmi"           "smoking"       "htn"           "diabetes"
#> [17] "ineligible"
```

```
names(dat.ineligible.stacked)
#>  [1] "studyid"       "survey.weight" "psu"           "strata"
#>  [5] "cvd"           "rheumatoid"    "age"           "sex"
#>  [9] "education"     "race"          "income"        "bmi"
#> [13] "smoking"       "htn"           "diabetes"      "ineligible"
#> [17] ".imp"
```

As we can see, the variable names are different in the two datasets. Particularly, `arthritis` and `age.cat` variables are not available in the `dat.ineligible.stacked` dataset. Now we will recode these variables in the same format as done for `impdata`:

```r
# Categorical age
summary(dat.ineligible.stacked$age)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    0.00    5.00   12.00   23.25   41.00   80.00
dat.ineligible.stacked$age.cat <-
  with(dat.ineligible.stacked,
       ifelse(age >= 20 & age < 50, "20-49",
              ifelse(age >= 50 & age < 65, "50-64",
                     ifelse(age >= 65, "65+", NA))))
dat.ineligible.stacked$age.cat <-
  factor(dat.ineligible.stacked$age.cat,
         levels = c("20-49", "50-64", "65+"))
```

Note that, we are assigning anyone with less than 20 age as missing.

```r
table(dat.ineligible.stacked$age.cat, useNA = "always")
#>
#> 20-49 50-64   65+  <NA>
#>  1100  2360  3430 18425
# Recode arthritis
dat.ineligible.stacked$arthritis <-
  car::recode(dat.ineligible.stacked$rheumatoid,
  " 'No' = 'No arthritis'; 'Yes' = 'Rheumatoid arthritis' ",
  as.factor = T)
```

Note: In the above step, we could also create two variables with missing values rather than recoding. The reason is that we will subset the design; no matter whether we recode or create missing values for ineligible, the only information we need from ineligible subjects is their survey features when creating the design.

The next step is to combine these two datasets (`impdata` and `dat.ineligible.stacked`).

```r
# Variable names in the imputed dataset
vars <- names(impdata)

# Set up the dataset for ineligible - same variables as impdata
dat.ineligible.stacked <- dat.ineligible.stacked[, vars]
```

Now we will merge ineligible and eligible subjects to make the full dataset of $5 \times 9{,}254 = 46{,}270$ subjects.

```
impdata2 <- rbind(impdata, dat.ineligible.stacked)
impdata2 <- impdata2[order(impdata2$.imp, impdata2$studyid),]
dim(impdata2)
#> [1] 46270    17
```

0.0.0.0.4 * 1.4 Prepating Survey design and subpopulation of eligible

The next step is to create the design on full dataset [with eligible (imputed) and ineligible (unimputed) subjects] of $5 \times 9{,}254 = 46{,}270$ subjects and subset the design for $5 \times 4{,}716 = 23{,}580$ subjects.

```
m <- 5
allImputations <- imputationList(lapply(1:m, function(n)
  subset(impdata2, subset=.imp==n)))

# Step 2: Survey data analysis
w.design0 <- svydesign(ids = ~psu,
                       weights = ~survey.weight,
                       strata = ~strata,
                      data = allImputations,
                      nest = TRUE) # Design on full data
w.design <- subset(w.design0, ineligible == 0) # Subset the design
```

We can see the length of the subsetted design:

```
dim(w.design)
#> [1] 4191   17    5
```

The subsetted design contains 4,191 subjects with 17 variables and 5 imputed datasets. Now we will run the design-adjusted logistic regression on and pool the estimate using Rubin's rule:

Step 2: Design adjusted regression analysis

```r
# Design-adjusted logistic regression
fit <- with(w.design,
            svyglm(I(cvd == "Yes") ~ arthritis + age.cat +
                     sex + education + race + income + bmi +
                     smoking + htn + diabetes,
                   family = quasibinomial))
res <- exp(as.data.frame(cbind(coef(fit[[1]]),
      coef(fit[[2]]),
      coef(fit[[3]]),
      coef(fit[[4]]),
      coef(fit[[5]]))))
names(res) <- paste("OR from m =", 1:5)
round(t(res),2)
#>               (Intercept) arthritisRheumatoid arthritis age.cat50-64 age.cat65+
#> OR from m = 1        0.02                         2.03         4.71      13.08
#> OR from m = 2        0.02                         2.04         4.74      13.21
#> OR from m = 3        0.02                         2.12         4.58      11.88
#> OR from m = 4        0.02                         2.10         4.59      12.45
#> OR from m = 5        0.02                         2.07         4.47      12.25
#>               sexFemale educationHigh school educationCollege graduate or above
#> OR from m = 1      0.46                 0.75                              0.75
#> OR from m = 2      0.45                 0.76                              0.79
#> OR from m = 3      0.47                 0.71                              0.70
#> OR from m = 4      0.46                 0.70                              0.71
#> OR from m = 5      0.45                 0.75                              0.77
#>               raceBlack raceHispanic raceOthers income$20,000 to $74,999
#> OR from m = 1      0.96         0.54       0.86                     0.48
#> OR from m = 2      0.97         0.54       0.88                     0.48
#> OR from m = 3      0.98         0.54       0.85                     0.54
#> OR from m = 4      0.96         0.53       0.86                     0.51
#> OR from m = 5      0.99         0.55       0.87                     0.53
#>               income$75,000 and Over  bmi smokingPrevious smoker
#> OR from m = 1                   0.33 1.02                   1.55
#> OR from m = 2                   0.32 1.03                   1.55
#> OR from m = 3                   0.36 1.01                   1.59
#> OR from m = 4                   0.34 1.02                   1.56
#> OR from m = 5                   0.34 1.02                   1.55
#>               smokingCurrent smoker htnYes diabetesYes
```

```
#> OR from m = 1                    1.43   1.35        3.01
#> OR from m = 2                    1.47   1.30        3.01
#> OR from m = 3                    1.45   1.48        3.13
#> OR from m = 4                    1.44   1.46        3.03
#> OR from m = 5                    1.45   1.47        3.05
```

Step 3: Pooling estimates

```
# Step 3: Pooled estimates
pooled.estimates <- MIcombine(fit)
OR <- round(exp(pooled.estimates$coefficients), 2)
OR <- as.data.frame(OR)
CI <- round(exp(confint(pooled.estimates)), 2)
OR <- cbind(OR, CI)
OR
#>                                      OR 2.5 % 97.5 %
#> (Intercept)                        0.02  0.01   0.05
#> arthritisRheumatoid arthritis      2.07  1.27   3.38
#> age.cat50-64                       4.62  2.41   8.85
#> age.cat65+                        12.56  7.60  20.76
#> sexFemale                          0.46  0.35   0.61
#> educationHigh school               0.73  0.42   1.28
#> educationCollege graduate or above 0.74  0.41   1.33
#> raceBlack                          0.97  0.67   1.42
#> raceHispanic                       0.54  0.28   1.03
#> raceOthers                         0.86  0.44   1.69
#> income$20,000 to $74,999           0.51  0.33   0.77
#> income$75,000 and Over             0.34  0.21   0.53
#> bmi                                1.02  0.99   1.05
#> smokingPrevious smoker             1.56  1.22   2.00
#> smokingCurrent smoker              1.45  1.02   2.04
#> htnYes                             1.41  0.78   2.54
#> diabetesYes                        3.05  1.98   4.70
```

Conclusion

Among US adults aged 20 years or more, the odds of CVD was approximately twice among those with rheumatoid arthri-

tis than no arthritis.

References

MCAR tests

MCAR tests are essential tools in the data analysis process. They help researchers understand the nature of missingness in their datasets, guide appropriate imputation strategies, and ensure the validity and reliability of statistical analyses.

MCAR data

In the initial chunk, several packages are loaded. These packages provide functions and tools necessary for the subsequent analysis, including multiple imputation, statistical modeling, and data visualization.

```r
# Load required packages
require(mice)
require(mitools)
require(survey)
require(remotes)
require(simcausal)
```

Data generating process

A Directed Acyclic Graph (DAG) is defined, which represents a causal model of how different variables in the dataset relate to each other. This DAG is used to simulate data based on the relationships defined. We generate `L` as a function of `P` and `B`.

```r
require(simcausal)
D <- DAG.empty()
D <- D +
  node("B", distr = "rnorm", mean = 0, sd = 1) +
```

```
  node("P", distr = "rnorm", mean = 0, sd = .7) +
  node("L", distr = "rnorm", mean = 2 + 2 * P + 3 * B, sd = 3) +
  node("A", distr = "rnorm", mean = 0.5 + L + 2 * P, sd = 1) +
  node("Y", distr = "rnorm", mean = 1.1 * L + 1.3 * A + B + 2 * P, sd = .5)
Dset <- set.DAG(D)
```

Generate DAG

The previously defined DAG is visualized, providing a graphical representation of the relationships between variables.

```
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
```

Generate Data

Using the DAG, a dataset is simulated. This dataset will be used for subsequent analysis.

```
Obs.Data <- sim(DAG = Dset, n = 10000, rndseed = 123)
head(Obs.Data)
#>   ID           B           P           L           A           Y
```

```
#> 1  1 -0.56047565  1.6595077  1.1286981  4.7541070  10.421710
#> 2  2 -0.23017749 -0.1167684  0.4142118  0.9388224   1.573070
#> 3  3  1.55870831  0.6488730  1.6633266  2.9227600   8.231805
#> 4  4  0.07050839 -0.3977062 -3.5873099 -5.0617852 -11.283620
#> 5  5  0.12928774  0.1575631 -0.5908993  1.1248743   1.907822
#> 6  6  1.71506499  0.7923901  3.7331116  5.8016423  14.839085
Obs.Data.original <- Obs.Data
```

Randomly set some data to missing

Some values in the dataset are randomly set to missing (i.e., randomly assign some L values to missing). This simulates a scenario where data might be missing completely at random (MCAR).

```
set.seed(123)
Obs.Data$L[sample(1:length(Obs.Data$L), size = 1000)] <- NA
summary(Obs.Data$L)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#> -17.116  -1.096   1.896   1.968   4.996  20.129    1000
```

The missing data patterns in the dataset are visualized. This helps in understanding which variables have missing values and how they might be related.

```
require(VIM)
res <- aggr(Obs.Data, plot = FALSE)
plot(res, numbers = TRUE, prop = FALSE)
```

Visualize via margin plots

Margin plots are used to compare the distributions of variables when a particular variable is missing versus when it is observed. This provides insights into how missingness might be related to the values of other variables.

- The red boxplot depicts the distribution of a variable in the data where **L has a missing value**.
- The blue boxplot depicts the distribution of the values of a variable in the data where **L has an observed value**.
- **Same median and spread (range) may mean no difference in the distribution**.

```
marginplot(Obs.Data[,c("L", "P")])
```

```
marginplot(Obs.Data[,c("L", "B")])
```

```
marginplot(Obs.Data[,c("L", "A")])
```

```r
marginplot(Obs.Data[,c("L", "Y")])
```

Little's MCAR test

A statistical test is conducted to determine if data is missing entirely at random (MCAR). The outcome of this test offers a deeper understanding of the reasons behind the missing data.

Little's 1988 chi-squared test evaluates if data is MCAR by checking for significant differences in the means of various

missing-value patterns (R. J. Little 1988). Based on the test's statistic and p-value, we can infer if the data is MCAR. The null hypothesis for this test is that the data is MCAR.

- The essence of this test is to compare means across groups with different missing data patterns. It uses a likelihood ratio test and assumes the data follows a multivariate normal distribution.
- If this test is rejected (indicated by a low p-value or a high statistic), it suggests the data might not be MCAR.

However, this test has several limitations (rdrr 2023):

- The test doesn't pinpoint which specific variables don't adhere to MCAR, meaning it doesn't highlight potential variables associated with missingness.
- It assumes multivariate normality. If this assumption isn't met, especially with non-normal or categorical variables, the test might not be reliable unless there's a large sample size.
- The test assumes that all missing data patterns have the same covariance matrix. This means it can't detect deviations from MCAR that are based on covariance, especially if the data is Missing at Random (MAR) or Missing Not at Random (MNAR).
- Research has shown that Little's MCAR test might have low power, especially when few variables don't follow MCAR, the association between the data and its missingness is weak, or if the data is MNAR.
- The test can only reject the MCAR assumption but can't confirm it. A non-significant result doesn't necessarily confirm that the data is MCAR.
- Even if the test result is significant, it doesn't rule out the possibility of the data being MNAR.

```r
require(naniar)
mcar_test(Obs.Data)
#> # A tibble: 1 x 4
#>   statistic    df p.value missing.patterns
#>       <dbl> <dbl>   <dbl>            <int>
#> 1      2.60     5   0.761                2
```

```r

require(misty)
na.test(Obs.Data)
#>  Little's MCAR Test
#>
#>       n nIncomp nPattern chi2 df  pval
#>   10000    1000        2 2.60  5 0.761
```

MCAR and normality test

Hawkins, in 1981, introduced a method to assess both multivariate normality and the consistency in variances, known as homoscedasticity. This method not only checks for consistent variances but also for mean equality (Hawkins 1981).

In 2010, Jamshidian and Jalal suggested a technique to compare covariances among groups with the same missing data patterns. They utilized the Hawkins test for data assumed to be normal and a non-parametric approach for other data types (Jamshidian and Jalal 2010).

The following package (Jamshidian and Jalal 2010) tests multivariate normality and homoscedasticity in the context of missing data.

```r
#library(devtools)
#install_github("cran/MissMech")
library(MissMech)
test.result <- TestMCARNormality(data = Obs.Data)
test.result
#> Call:
#> TestMCARNormality(data = Obs.Data)
#>
#> Number of Patterns:  2
#>
#> Total number of cases used in the analysis:  10000
#>
#>  Pattern(s) used:
#>           ID   B   P    L   A   Y   Number of cases
#> group.1    1   1   1   NA   1   1              1000
#> group.2    1   1   1    1   1   1              9000
```

```
#>
#>
#>     Test of normality and Homoscedasticity:
#>   ----------------------------------------
#>
#> Hawkins Test:
#>
#>     P-value for the Hawkins test of normality and homoscedasticity:  1.749746e-15
#>
#>     Either the test of multivariate normality or homoscedasticity (or both) is rejected.
#>     Provided that normality can be assumed, the hypothesis of MCAR is
#>     rejected at 0.05 significance level.
#>
#> Non-Parametric Test:
#>
#>     P-value for the non-parametric test of homoscedasticity:  0.4019219
#>
#>     Reject Normality at 0.05 significance level.
#>     There is not sufficient evidence to reject MCAR at 0.05 significance level.
summary(test.result)
#>
#> Number of imputation:  1
#>
#> Number of Patterns:  2
#>
#> Total number of cases used in the analysis:  10000
#>
#>  Pattern(s) used:
#>           ID   B   P    L   A   Y   Number of cases
#> group.1    1   1   1   NA   1   1              1000
#> group.2    1   1   1    1   1   1              9000
#>
#>
#>     Test of normality and Homoscedasticity:
#>   ----------------------------------------
#>
#> Hawkins Test:
#>
#>     P-value for the Hawkins test of normality and homoscedasticity:  1.749746e-15
#>
```

```
#> Non-Parametric Test:
#>
#>     P-value for the non-parametric test of homoscedasticity:  0.4019219
```

```r
png("E:/GitHub/EpiMethods/Images/missingdata/boxplot.png", width = 600, height = 600)
boxplot(test.result)
dev.off()
```

```r
boxplot(test.result)
```

Boxplots of p-values corresponding to each set of the missing data patterns for the Neyman test of Uniformity
P-value
0.00
0.05
0.10
0.15
1
2
Missing data pattern group

Boxplots of the T-value test statistics corresponding to each set of missing data patterns for the non-parametric test
T_i
0.1
0.3
0.5
0.7
1
2
Missing data pattern group

Non-MCAR data

Set some data to missing based on a rule

In this section, the original dataset is restored. Then, for a specific column, any value greater than a certain threshold is set to 'missing'. A summary of this column is then provided to understand the distribution of missing values.

```
Obs.Data <- Obs.Data.original
Obs.Data$L[Obs.Data$L > 7.79] <- NA
summary(Obs.Data$L)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#> -17.116  -1.482   1.330   1.050   3.954   7.787    1035
```

The dataset's missing data patterns are visualized. This visualization helps in understanding which data points are missing and their distribution across the dataset.

```
res <- aggr(Obs.Data, plot = FALSE)
plot(res, numbers = TRUE, prop = FALSE)
```

Visualize via margin plots

Margin plots are used to visualize the relationship between two variables, especially when one of them has missing values. Here, the relationship of a specific column with missing values is visualized against other columns in the dataset. This helps in understanding how the missingness in one variable might relate to other variables.

```
marginplot(Obs.Data[,c("L", "P")])
```

```
marginplot(Obs.Data[,c("L", "B")])
```

```r
marginplot(Obs.Data[,c("L", "A")])
```

```r
marginplot(Obs.Data[,c("L", "Y")])
```

Little's MCAR test

Little's MCAR test is applied to the dataset to check if the data is missing completely at random. This test provides a statistic and a p-value to determine the nature of missingness in the data.

```
mcar_test(Obs.Data)
#> # A tibble: 1 x 4
#>   statistic    df p.value missing.patterns
#>       <dbl> <dbl>   <dbl>            <int>
#> 1     3511.     5       0                2
na.test(Obs.Data)
#>  Little's MCAR Test
#>
#>       n nIncomp nPattern    chi2 df  pval
#>   10000    1035        2 3511.08  5 0.000
```

MCAR and normality test

A test is conducted to check if the data follows a multivariate normal distribution and if the variances across different groups are consistent (homoscedasticity). The results of this test, along with a summary and a boxplot visualization, are

provided to understand the distribution and characteristics of the data.

```r
test.result <- TestMCARNormality(data = Obs.Data)
test.result
#> Call:
#> TestMCARNormality(data = Obs.Data)
#>
#> Number of Patterns:  2
#>
#> Total number of cases used in the analysis:  10000
#>
#>  Pattern(s) used:
#>           ID   B   P    L   A   Y   Number of cases
#> group.1    1   1   1    1   1   1              8965
#> group.2    1   1   1   NA   1   1              1035
#>
#>
#>     Test of normality and Homoscedasticity:
#>   ------------------------------------------
#>
#> Hawkins Test:
#>
#>     P-value for the Hawkins test of normality and homoscedasticity:  2.401873e-30
#>
#>     Either the test of multivariate normality or homoscedasticity (or both) is rejected.
#>     Provided that normality can be assumed, the hypothesis of MCAR is
#>     rejected at 0.05 significance level.
#>
#> Non-Parametric Test:
#>
#>     P-value for the non-parametric test of homoscedasticity:  0
#>
#>     Hypothesis of MCAR is rejected at  0.05 significance level.
#>     The multivariate normality test is inconclusive.
summary(test.result)
#>
#> Number of imputation:  1
#>
#> Number of Patterns:  2
#>
```

```
#> Total number of cases used in the analysis:  10000
#>
#>  Pattern(s) used:
#>           ID   B   P    L   A   Y   Number of cases
#> group.1    1   1   1    1   1   1              8965
#> group.2    1   1   1   NA   1   1              1035
#>
#>
#>     Test of normality and Homoscedasticity:
#>   ----------------------------------------
#>
#> Hawkins Test:
#>
#>     P-value for the Hawkins test of normality and homoscedasticity:  2.401873e-30
#>
#> Non-Parametric Test:
#>
#>     P-value for the non-parametric test of homoscedasticity:  0
```

```r
png("E:/GitHub/EpiMethods/Images/missingdata/boxplot1.png", width = 600, height = 600)
boxplot(test.result)
dev.off()
```

```r
boxplot(test.result)
```

Boxplots of p-values corresponding to each set of the missing data patterns for the Neyman test of Uniformity
P-value
0.00
0.05
0.10
0.15
1
2
Missing data pattern group

Boxplots of the T-value test statistics corresponding to each set of missing data patterns for the non-parametric test
T_i
0.1
0.3
0.5
0.7
1
2
Missing data pattern group

Effect modification

In this tutorial, we delve into the concept of effect modification.

> **❗ Important**
>
> We discussed about effect modification in an earlier discussion.

We start by loading the necessary packages that will aid in the analysis. We also define a function `tidy.pool_mi` to streamline the pooling process for multiple imputation results.

```r
# Load required packages
library(survey)
require(interactions)
require(mitools)
require(mice)
require(miceadds)
library(modelsummary)

tidy.pool_mi <- function(x, ...) {
  msg <- capture.output(out <- summary(x, ...))
  out$term <- row.names(out)
  colnames(out) <- c("estimate", "std.error", "statistic", "p.value",
                     "conf.low", "conf.high", "miss", "term")
  return(out)
}
```

Data

We load a dataset named `smi`. This dataset is a list of multiple imputed datasets, which is evident from the structure and the way we access its elements.

```r
require(mitools)
data(smi)
length(smi)
#> [1] 2
length(smi[[1]])
#> [1] 5
head(smi[[1]][[1]])
#>       id wave mmetro parsmk         drkfre         alcdos alcdhi           smk
#> 1 920006    1      1      0    Non drinker    Non drinker      0 non/ex-smoker
#> 2 920006    2      1      0    Non drinker    Non drinker      0 non/ex-smoker
#> 3 920006    3      1      0    Non drinker    Non drinker      0 non/ex-smoker
#> 4 920006    4      1      0    Non drinker    Non drinker      0 non/ex-smoker
#> 5 920006    5      1      0    Non drinker    Non drinker      0 non/ex-smoker
#> 6 920006    6      1      0 not in last wk not in last wk      0 non/ex-smoker
#>   cistot mdrkfre sex drinkreg
#> 1      0       0   1    FALSE
#> 2      0       0   1    FALSE
#> 3      0       0   1    FALSE
#> 4      1       0   1    FALSE
#> 5      4       0   1    FALSE
#> 6      3       0   1    FALSE
```

Model with interaction and ORs

We're interested in understanding how the variable `wave` interacts with `sex` in predicting `drinkreg`. We fit two logistic regression models, one for each level of the `sex` variable (0 males, 1 females), to understand this interaction. `wave` is the exposure variable here.

For effect modifier sex = 0 (males)

```r
models <- with(smi, glm(drinkreg~ wave + sex + wave*sex, family = binomial()))
summary(pool(models, rule = "rubin1987"), conf.int = TRUE, exponentiate = TRUE)[2,]
#>   term estimate  std.error statistic       df      p.value   2.5 %   97.5 %
#> 2 wave 1.271952 0.06587423  3.651693 234.4974 0.0003212903 1.11714 1.448217
```

For effect modifier sex = 1 (females) (just changing reference)

```
models2<-with(smi, glm(drinkreg~ wave + I(sex==0) + wave*I(sex==0),family=binomial()))
summary(pool(models2, rule = "rubin1987"),conf.int = TRUE, exponentiate = TRUE)[2,]
#>   term estimate  std.error statistic       df      p.value    2.5 %   97.5 %
#> 2 wave 1.225438 0.05876369   3.45959 240.3053 0.0006399352 1.091487 1.375828
```

- Notice the ORs for `wave` in the above 2 analyses. These are basically our target.
- For proper survey data analysis, you will have to work with design and make sure you subset your subpopulation (those eligible) appropriately.

Simple slopes analyses

We perform a simple slopes analysis for each imputed dataset. This analysis helps in understanding the relationship between the predictor and the outcome at specific levels of the moderator.

```
a1 <- sim_slopes(models[[1]], pred = wave, modx = sex)
a2 <- sim_slopes(models[[2]], pred = wave, modx = sex)
a3 <- sim_slopes(models[[3]], pred = wave, modx = sex)
a4 <- sim_slopes(models[[4]], pred = wave, modx = sex)
a5 <- sim_slopes(models[[5]], pred = wave, modx = sex)
```

After obtaining the results from each imputed dataset, we pool them to get a consolidated result. This is done separately for each level of the `sex` variable.

Pooled results for sex = 0

```
# For sex = 0
ef.lev <- 1
est <- c(a1$slopes$Est.[ef.lev],
         a2$slopes$Est.[ef.lev],
```

```r
         a3$slopes$Est.[ef.lev],
         a4$slopes$Est.[ef.lev],
         a5$slopes$Est.[ef.lev])
se <- c(a1$slopes$S.E.[ef.lev],
        a2$slopes$S.E.[ef.lev],
        a3$slopes$S.E.[ef.lev],
        a4$slopes$S.E.[ef.lev],
        a5$slopes$S.E.[ef.lev])
vr <- se^2
OR <- exp(est)
OR.se <- OR * se
OR.v <- OR.se^2

mod_pooled <- miceadds::pool_mi(qhat=OR, u=OR.v)
tidy.pool_mi(mod_pooled)
#>   estimate  std.error statistic      p.value conf.low conf.high   miss term
#> 1 1.272164 0.08386552  15.16909 6.987679e-39 1.107118   1.43721 12.2 %    1
summary(MIcombine(as.list(OR), as.list(OR.v)))
#> Multiple imputation results:
#>       MIcombine.default(as.list(OR), as.list(OR.v))
#>    results         se   (lower  upper) missInfo
#> 1 1.272164 0.08386552 1.107118 1.43721     12 %
```

Pooled results for sex = 1

```r
# For sex = 1
ef.lev <- 2
est <- c(a1$slopes$Est.[ef.lev],
         a2$slopes$Est.[ef.lev],
         a3$slopes$Est.[ef.lev],
         a4$slopes$Est.[ef.lev],
         a5$slopes$Est.[ef.lev])
se <- c(a1$slopes$S.E.[ef.lev],
        a2$slopes$S.E.[ef.lev],
        a3$slopes$S.E.[ef.lev],
        a4$slopes$S.E.[ef.lev],
        a5$slopes$S.E.[ef.lev])
vr <- se^2
```

```r
OR <- exp(est)
OR.se <- OR * se
OR.v <- OR.se^2

mod_pooled <- miceadds::pool_mi(qhat=OR, u=OR.v)
tidy.pool_mi(mod_pooled)
#>   estimate  std.error statistic      p.value conf.low conf.high   miss term
#> 1 1.225598 0.07204679  17.01113 2.282427e-46 1.083838  1.367357 11.9 %    1
summary(MIcombine(as.list(OR), as.list(OR.v)))
#> Multiple imputation results:
#>       MIcombine.default(as.list(OR), as.list(OR.v))
#>    results         se   (lower   upper) missInfo
#> 1 1.225598 0.07204679 1.083838 1.367357     12 %
```

R functions (M)

The list of new R functions introduced in this *Missing data analysis* lab component are below:

Function_name	Package_name	Use
aggr	VIM	To calculate/plot the missing values in the variables
boxplot	base/graphics	To produce a box plot
bwplot	mice	To produce box plot to compare the imputed and observed values
colMeans	base	To compute the column-wise mean, i.e., mean for each variable/column
complete	mice	To extract the imputed dataset
complete.cases	base/stats	To select the complete cases, i.e., observations without missing values
D1	mice	To conduct the multivariate Wald test with D1-statistic
densityplot	mice	To produce desnsity plots
expression	base	To set/create an expression
imputationList	mice	To combine multiple imputed datasets
marginplot	VIM	To draw a scatterplot with additional information when there are missi
mcar_test	naniar	To conduct Little's MCAR test
md.pattern	mice	To see the pattern of the missing data
mice	mice	To impute missing data where the argument m represents the number o
MIcombine	mitools	To combine/pool the results using Rubin's rule
MIextract	mitools	To extract parameters from a list of outputs
na.test	misty	To conduct Little's MCAR test
parlmice	mice	To run ‘mice‘ function in parallel, i.e., parallel computing of mice
plot_missing	DataExplorer	To plot the profile of missing values, e.g., the percentage of missing per
pool	mice	To pool the results using Rubin's rule
pool.compare	mice	To compare two nested models
pool_mi	miceadds	To combine/pool the results using Rubin's rule
quickpred	mice	To set imputation model based on the correlation
sim_slopes	interactions	To perform simple slope analyses
TestMCARNormality	MissMech	To test multivariate normality and homoscedasticity in the context of m
unlist	base	To convert a list to a vector

Quiz (M)

> ⚠ Warning
>
> We are currently in the beta testing phase for our quiz page updates, so please be aware that some responses may not be accurate. Thank you for your patience.
> For those interested in downloading the quiz questions, you'll find a section below dedicated to downloading.

Live Quiz

You can click on the live quiz options to either see hints of why your response may be incorrect (response marked in red), or whether your choice of response is correct (response marked in green):

Download Quiz

1. **Downloading the File:**
   - Navigate to the link: See here.
   - Right-click on the link and select "Save link as…" from the dropdown menu.
   - Alternatively click here for downloading the quizzes.
   - Choose a destination folder on your computer where you'd like to save the file (e.g., Desktop). **Remember this location**, as you'll need to navigate to it later.
2. **Setting Up RStudio:**

- If you don't have RStudio installed, see the download link in here.
- Launch RStudio after installing.

3. **Installing Necessary R Packages:**

   - Before running the R Markdown file, ensure you have the required packages. In RStudio's console (located at the bottom left by default), enter the following commands to install them:

```r
install.packages("learnr")
install.packages("xfun")
```

4. **Opening and Running the File:**

   - In RStudio, go to `File` > `Open File...`.
   - Navigate to the folder where you saved the `Rmd` file and select it to open.
   - Once the file is open in RStudio, you'll see a "Run Document" button (green) at the top of the script editor. Click on it to run the R Markdown Quiz file.

App (M)

Below is an example of an app that utilizes the NHANES dataset following the tutorial materials. Users can visualize the results from various imputations as well as pooled results from multiple imputation.

> Tip
>
> **Web App may need some time to load the necessary files, depending on your internet speed.**

> Important
>
> If you want to run the app locally in your computer instead for faster load time: you can download the app directly in your R via `shiny::runGitHub("EpiShinyLiveM", "ehsanx", subdir = "app")`. Make sure that `shiny`, and `ggplot2` packages are installed in your computer/R.

> Warning
>
> **Bug Report**:
> Fill out this form to report any issues with the app.

Exercise (M)

> Tip
>
> You can download all of the related files in a zip file **missingdataEx.zip** from Github folder, or just by clicking this link directly.
>
> - Navigate to the GitHub folder (above link) where the ZIP file is located.
> - Click on the file name (above zip file) to open its preview window.
> - Click on the Download button to download the file. If you can't see the Download button, click on "Download Raw File" link that should appear on the page.

Problem Statement

We will use the article by Williams AR, Wilson-Genderson M, Thomson MD. (2021)

We will reproduce some results from the article. The authors used NHANES 2015-16 and 2017-18 datasets to create their analytic dataset. The combined dataset contains 19,225 subjects with 20 relevant variables for this exercise:

Survey information

- id: Respondent sequence number
- survey.weight: Full sample 4 year interview weight
- psu: Masked pseudo PSU
- strata: Masked pseudo strata (strata is nested within PSU)

4 Outcome variables

- weight.loss.behavior: doing lifestyle behavior changes - controlling or losing weight
- exercise.behavior: doing lifestyle behavior changes - increasing exercise
- salt.behavior: doing lifestyle behavior changes - reducing salt in diet
- fat.behavior: doing lifestyle behavior changes - reducing fat in diet

4 predictors (i.e., exposure variables)

- weight.loss.advice: told by a doctor or health professional - to control/lose weight
- exercise.advice: told by a doctor or health professional - to exercise
- salt.advice: told by a doctor or health professional - to reduce salt in diet
- fat.advice: told by a doctor or health professional - to reduce fat/calories

Confounders and other variables

- gender: Gender
- age: Age in years at screening
- income: The ratio of family income to federal poverty level
- race: Race/Ethnicity
- bmi: Body Mass Index in kg/m^2
- comorbidity: Comorbidity index
- DIQ010: Self-report to have been informed by a provider to have diabetes
- BPQ020: Self-report to have been informed by a provider to have hypertension

Question 1: Analytic dataset

1(a) Importing dataset

```r
# download the data in the same folder
load("Data/missingdata/Williams2021.RData")
```

1(b) Subsetting according to eligibility

Create a dataset with missing values in outcomes, predictors, and confounders. As shown in Figure 1, the sample size should be 4,746.

```r
# Drop < 18 years
dat <- dat.full
dat <- dat[dat$age >= 18,]

# Eligibility
dat <- dat[dat$DIQ010=="Yes" | dat$BPQ020=="Yes",]

# Dataset with missing values in outcomes, predictors, and confounders
dat.with.miss <- dat
nrow(dat.with.miss) # N = 4,746
#> [1] 4746
```

1(c) Dataset with missing values only in confounders

Create a dataset with missing values in only in confounders. There should not be any missing values in the outcomes or predictors. As shown in Figure 1, the sample size should be 4,716.

- Hint: there are four outcome variables and four predictors in this paper. Read the "Self-reported behavior change and receipt of advice" paragraph.

```r
dat <- dat.with.miss

# Drop missing or don't know outcomes
dat <- dat[complete.cases(dat$weight.loss.behavior),]
dat <- dat[complete.cases(dat$exercise.behavior),]
dat <- dat[complete.cases(dat$salt.behavior),]
dat <- dat[complete.cases(dat$fat.behavior),]

# Drop missing or don't know predictors
dat <- dat[complete.cases(dat$weight.loss.advice),]
dat <- dat[complete.cases(dat$exercise.advice),]
dat <- dat[complete.cases(dat$salt.advice),]
dat <- dat[complete.cases(dat$fat.advice),]

# Dataset without missing in outcomes and predictors but missing in confounders
dat.with.miss2 <- dat
nrow(dat.with.miss2) # N = 4,716
#> [1] 4716
```

1(d) Reproduce Table 1

Create the first column of Table 1 of the article.

- Hint 1: The authors reported unweighted frequencies, and thus, survey features should not be utilized to answer this question. Use `tableone` package.
- Hint 2: You may need to generate the `Condition` variable.
- Hint 3: `age` and `comorbidity` are numerical variables. `tableone` package gives mean (SD) for numerical variables by default. For this exercise, instead of reporting the frequency, you could report the mean (SD) for `age` and `comorbidity`.

```r
dat <- dat.with.miss2

# Create the condition variable
dat$condition <- NA
```

```r
dat$condition[dat$BPQ020 == "Yes"] <- "Hypertension Only"
dat$condition[dat$DIQ010 == "Yes"] <- "Diabetes Only"
dat$condition[dat$BPQ020 == "Yes" & dat$DIQ010 == "Yes"] <- "Both"
dat$condition <- factor(dat$condition, levels=c("Hypertension Only", "Diabetes Only",
                                                "Both"))
table(dat$condition, useNA = "always")
#>
#> Hypertension Only     Diabetes Only              Both              <NA>
#>              3004               533              1179                 0
```

```r
# First column of Table 1
vars <- c("gender", "age", "income", "race", "bmi", "condition", "comorbidity")
tab1 <- CreateTableOne(vars = vars, data = dat, includeNA = F)
print(tab1, format = "f")
#>
#>                            Overall
#>   n                         4716
#>   gender = Male             2332
#>   age (mean (SD))          59.94 (14.96)
#>   income
#>      <100%                   881
#>      100-199%               1193
#>      200-299%                672
#>      300-399%                424
#>      400+%                   930
#>   race
#>      Hispanic               1161
#>      Non-Hispanic white     1630
#>      Non-Hispanic black     1239
#>      Others                  686
#>   bmi
#>      Reference               753
#>      Overweight             1372
#>      Obese                  2287
#>   condition
#>      Hypertension Only      3004
#>      Diabetes Only           533
#>      Both                   1179
#>   comorbidity (mean (SD))  1.29 (1.45)
```

Question 2: Dealing with missing values in confoudners [100% grade]

2(a) Check missingness using a plot

In the dataset created in 1(c), use a plot to check missingness. In the plot, include only the outcome variables, predictors, and confounders.

- Hint 1: There are four outcome variables and four predictor variables used in the study.
- Hint 2: The authors considered the following confounders: gender, age, income, race, bmi, condition, and comorbidity.

```r
# Create the condition variable in the analytic dataset
dat.with.miss2$condition[dat.with.miss2$BPQ020 == "Yes"] <- "Hypertension Only"
dat.with.miss2$condition[dat.with.miss2$DIQ010 == "Yes"] <- "Diabetes Only"
dat.with.miss2$condition[dat.with.miss2$BPQ020 == "Yes" &
                         dat.with.miss2$DIQ010 == "Yes"] <- "Both"
dat.with.miss2$condition <- factor(dat.with.miss2$condition,
                                   levels=c("Hypertension Only", "Diabetes Only", "Both"))

# Variables of interest
vars <- c(
  # Outcome
  "weight.loss.behavior", "exercise.behavior", "salt.behavior", "fat.behavior",

  #Predictors
  "weight.loss.advice", "exercise.advice", "salt.advice", "fat.advice",

  # Confounders
  "gender", "age", "income", "race", "bmi", "condition", "comorbidity")

# Plot missing values using DataExplorer
plot_missing(dat.with.miss2[,vars])
```

2(b) Reproduce Table 3: Multiple imputation

Perform multiple imputations to deal with missing values only in confounders. Use the dataset created in Dataset with missing values only in confounders (`dat.with.miss2`). Consider 5 imputed datasets, 5 iterations, and fit the design-adjusted logistic regression in all of the 5 imputed datasets. Obtain the pooled adjusted odds ratio with the 95% confidence intervals. In this case, consider only one outcome and one predictor that are related to lose weights, i.e., create only the **first column of Table 3**.

- Hint 1: Setup the data such that the variables are of appropriate types. `lapply` function could be helpful.
- Hint 2: Relevel the confounders as shown in Table 3.
- Hint 3: Use the strata variable as an auxiliary variable in the imputation model, but not the survey weight or PSU variable.
- Hint 4: Consider predictive mean matching method for bmi and comorbidity variable in the imputation model.
- Hint 5: Set your seed to 123.
- Hint 6: Remove any subject ID variable from the imputation model, if created in an intermediate step.
- Hint 7: The point and interval estimates could be slightly different than shown in Table 3. But they should very

close.

- Hint 8: Remember to keep count of the ineligible subjects from the full data, and consider adding them back in the imputed datasets (so that all the weight, strata and cluster information are available in the design).

```r
## Setup the data such that the variables are of appropriate types
factor.names <- c("weight.loss.behavior", "exercise.behavior", "salt.behavior",
                  "fat.behavior", "weight.loss.advice", "exercise.advice",
                  "salt.advice", "fat.advice", "gender", "income", "race", "bmi",
                  "condition")
# your codes


## Change the reference categories
# your codes


## Imputation model set up
# your codes


## Regression analysis
# your codes


## Pooled estimates
# your codes
```

Question 3: Dealing with missing values in outcome, predictor, and confoudners [optional]

Perform multiple imputations to deal with missing values only in outcome, predictor, confounders. Use the Multiple Imputation then deletion (MID) approach. Use the dataset created in Subsetting according to eligibility (`dat.with.miss`). Consider 5 imputed datasets, 5 iterations, and fit the design-adjusted

logistic regression in all of the 5 imputed datasets. Obtain the pooled adjusted odds ratio with the 95% confidence intervals. In this case, consider only one outcome and one predictor that are related to reduce fat/calories, i.e., create only the **fourth column of Table 3**.

- Hint 1: Setup the data such that the variables are of appropriate types.
- Hint 2: Relevel the confounders as shown in Table 3.
- Hint 3: Use the strata variable as an auxiliary variable in the imputation model, but not the survey weight or PSU variable.
- Hint 4: Include all 4 outcomes and 4 predictors in your imputation model.
- Hint 5: Consider predictive mean matching method for bmi and comorbidity variable in the imputation model.
- Hint 6: Set your seed to 123.
- Hint 7: Remove any subject ID variable from the imputation model, if created in an intermediate step.
- Hint 8: The point and interval estimates could be slightly different than shown in Table 3. But they should very close.
- Hint 9: Remember to keep count of the ineligible subjects from the full data, and consider adding them back in the imputed datasets (so that all the weight, strata and cluster information are available in the design).

```
## Create a missing indicator so that MID can be applied
# your codes here

## MID
# your codes here
```

Part VIII

Propensity score

Background

This chapter provides a comprehensive set of tutorials that guide readers through various methodologies of Propensity Score Matching (PSM) and Multiple Imputation (MI) using R, with practical applications using datasets like the Canadian Community Health Survey (CCHS) and the National Health and Nutrition Examination Survey (NHANES). The tutorials explore different scenarios and methodologies in handling and analyzing data, particularly focusing on estimating treatment effects and managing missing data. They delve into specific examples, such as exploring the relationship between Osteoarthritis (OA) and Cardiovascular Disease (CVD), and between Body Mass Index (BMI) and diabetes, while emphasizing the importance of accurate data handling, variable management, and robust analysis through PSM and MI. The tutorials are meticulously structured, providing step-by-step guides, code snippets, and thorough explanations, ensuring that readers can comprehend and replicate the processes in their research, thereby enhancing the reliability and robustness of their analyses, especially in the presence of missing data.

> **Note**
>
> Should you find yourself seeking a refresher on PSM, we invite you to revisit our dedicated/external tutorial, which elucidates PSM within a non-survey data analysis context. This resource not only provides a foundational understanding but also serves as a comprehensive guide through the nuanced steps of PSM. Additionally, our external discussion page offers a succinct summary of the tutorial and thoughtfully extends the conversation into more intricate directions, exploring the complexities and advanced applications of PSM (propensity score weighting, categorical and continuous exposure). Both resources are crafted to enhance your understanding and application of PSM, ensuring a robust and informed approach to your data analysis journey

Stepping into this chapter, we are diving deeper into the world of survey data analysis, exploring how to combine propensity score matching (PSM) and strategies for handling missing data. PSM helps us balance our data, making sure our study groups are comparable, while managing missing data ensures our results are as accurate as possible. In the upcoming tutorials, we will weave through the steps of using PSM while also dealing with the gaps in our data, ensuring our analyses are solid and dependable. So, this chapter is not just a next step, but a leap into a more advanced exploration, blending matching methods with careful data handling strategies.

> **!** Important
>
> **Datasets**:
> All of the datasets used in this tutorial can be accessed from this GitHub repository folder

Overview of tutorials

Covariate matching using CCHS: example of OA-CVD

The tutorial illustrate a comprehensive data analysis workflow using R, focusing on matching methods to estimate treatment effects with the CCHS data. Initially, we conduct data preprocessing steps to handle categorical variables and missing data. Subsequent sections delve into setting up design objects for survey-weighted analyses and conducting preliminary analyses to explore variable distributions and treatment effects. The core of the analysis involves implementing matching techniques, starting with a single variable and progressively including more variables to refine the matching. Various matching scenarios are explored, each followed by logistic regression models to estimate treatment effects.

Propensity score matching using CCHS: revisiting example of OA-CVD

The tutorial provides a thorough walkthrough of implementing Propensity Score Matching (PSM) in R, specifically in the context of an OA - CVD health study from the CCHS. PSM is utilized to mitigate bias from confounding variables in observational studies by pairing treated and control units with analogous propensity scores. The guide underscores that PSM is iterative, often requiring refinement of the matching strategy to achieve satisfactory covariate balance in the matched sample. Various strategies for estimating treatment effects in the matched sample are explored, each with distinct assumptions and implications. The tutorial also delves into different matching strategies, such as nearest-neighbor matching with

and without calipers, matching with different ratios, and matching with replacement, all while emphasizing the importance of assessing and re-assessing covariate balance at each step using both graphical and numerical methods.

Propensity score matching using NHANES: example of OA - CVD

The provided text outlines methodologies for conducting PSM using the NHANES dataset, with a particular emphasis on handling survey design and weights in the analysis. Three distinct approaches, attributed to Zanutto (2006), DuGoff et al. (2014), and Austin et al. (2018), are delineated, each with a structured four-step process: 1) specifying the propensity score model, 2) matching treated and untreated subjects based on estimated propensity scores, 3) comparing baseline characteristics between matched groups, and 4) estimating treatment effects using the matched sample. The procedures utilize various R packages and functions to manipulate data, visualize missing data patterns, format variables, and perform analyses, ensuring that survey weights and design are appropriately considered to avoid bias in population-level effect estimates. The text underscores the importance of incorporating survey design into at least propensity score outcome analysis (e.g., during step 4: treatment effect estimation), as neglecting survey weights can significantly impact the estimates of population-level effects.

Propensity score matching using NHANES: example of BMI - diabetes

The tutorial provides a comprehensive guide on implementing PSM in R, utilizing the NHANES dataset, with a specific focus on diabetes as an outcome and body mass index (BMI) as an exposure variable. The methodology encompasses ensuring accurate and reproducible results in PSM. The tutorial, again, meticulously follows three distinct approaches for PSM, as recommended by Zanutto (2006), DuGoff et al. (2014), and Austin et al. (2018), each providing a unique perspective on handling

and analyzing variables within the propensity score model. Notably, the tutorial introduces a nuanced approach to variable handling, model specifications, and matching steps, ensuring a thorough understanding of implementing PSM with varied methodologies. Furthermore, the tutorial introduces a "double adjustment" step in each approach, providing a robust estimate of the treatment effect while adjusting for covariates, thereby offering readers a holistic view on conducting PSM with a different set of variables and methodologies in the analysis steps.

Propensity score matching using NHANES when some variables have missing observations

This tutorial offers a clear and straightforward guide on how to use Propensity Score Matching (PSM) and Multiple Imputation (MI) in R, using the NHANES dataset for practical illustration. The main goal is to explore the relationship between "diabetes" (outcome) and being "born in the US" (exposure), while effectively managing missing data through MI. The first part of the tutorial, focusing on logistic regression, explains how to perform multiple imputations, fit a logistic regression model to all imputed datasets, and then obtain pooled Odds Ratios (OR) and 95% confidence intervals. Following this, the PSM analysis section carefully applies the PSM method, following Zanutto E. L. (2006), to all imputed datasets, and presents the pooled OR estimates and 95% confidence intervals. The tutorial emphasizes the crucial role of managing missing data through multiple imputation and provides a detailed, step-by-step guide, including code and thorough explanations, to ensure a deep understanding and ability to replicate the PSM with MI process in epidemiological research. This resource is invaluable for researchers and data analysts looking to strengthen their analyses when dealing with missing data.

Propensity score weighting

In this section, propensity score weighting with different methods is employed to estimate treatment effects in a modified

NHANES dataset. Three different propensity score weighting approaches are applied: Zanutto's (2006), DuGoff et al.'s (2014), and Austin et al.'s (2018). Each approach involves multiple steps, including the estimation of propensity scores, the calculation of weights (both unstabilized and stabilized), balance checking to ensure covariate balancing, and fitting outcome models using the weighted data. The double adjustment technique is also considered in each approach.

Propensity score matching with multiple imputation in subpopulations

In this section, the goal is to use propensity score matching (PSM) with multiple imputation (MI) to analyze a modified dataset from NHANES 2017-2018. The analysis focuses on specific subpopulations defined by eligibility criteria. This analysis provides insights into how to handle complex survey data with missing values and perform PSM with MI for subpopulations defined by eligibility criteria.

Propensity score weighting with multiple imputation in subpopulations

In this chapter, we employ propensity score (PS) weighting with MI to analyze a modified dataset from NHANES 2017-2018, focusing on specific subpopulations defined by eligibility criteria. Here we demonstrate how to perform PS weighting with MI for subpopulations defined by eligibility criteria.

Propensity score weighting for multiple treatment categories

In this chapter, we use propensity score weighting for multiple treatment categories using CCHS data.

Tip

Optional Content:
You'll find that some sections conclude with an optional video walkthrough that demonstrates the code. Keep in mind that the content might have been updated since these videos were recorded. Watching these videos is optional.

Warning

Bug Report:
Fill out this form to report any issues with the tutorial.

Concepts (S)

Propensity Score Analysis

This section provides a comprehensive exploration into various facets of propensity score (PS) methods and their application in observational studies and surveys. Beginning with an in-depth look into key concepts and calculations related to ATE and ATT, the content navigates through the practical application and diagnostic checks of covariate balance using the SMD. It further elucidates the methodology and application of PS, particularly focusing on matching and weighting to mitigate bias and create comparable groups for causal inference. The intricacies of employing PS methods within surveys are explored, highlighting different approaches and the incorporation of design variables in PS and outcome models. Fundamental assumptions for causal inference, namely Conditional Exchangeability, Positivity, and Causal Consistency, are dissected to form a foundational understanding for conducting robust causal analyses. Additionally, the content optionally delves into the nuances of implementing IPW in surveys. Lastly, additional optional content features an insightful workshop, offering more explanations of PS method implementations in research contexts.

Reading list

Key reference: (Peter C. Austin 2011)

Optional reading:

- Propensity score introduction (M. Karim 2021) External link

- Extensions of Propensity score approaches External link: prepared for Guest Lecture in SPPH 500/007 (Analytical Methods in Epidemiological Research)
- Propensity score for complex surveys External link: Uses the same lectures here, with some added text descriptions. This also includes a a structured framework for reporting analyses using PS methods in research manuscripts.
- Reporting guideline (Stuart 2018; Simoneau et al. 2022)
- Assumptions (Miguel A. Hernán and Robins 2020)

Theoretical references for propensity score analyses in complex surveys:

(Peter C. Austin, Jembere, and Chiu 2018; DuGoff, Schuler, and Stuart 2014; Zanutto 2006; Leite, Stapleton, and Bettini 2018; Lenis, Ackerman, and Stuart 2018; Lenis et al. 2017; Ridgeway et al. 2015)

Video Lessons

Target parameters

Average Treatment Effect (ATE) vs. Average Treatment effect on the Treated (ATT)

Balance

Balance and standardized mean difference (SMD) in observational studies

Propensity score matching

Propensity score matching in complex survey

Propensity score weighting in complex survey

Causal Assumptions

Conference Workshop (Optional)

Post Conference Workshop for 2021 Conference - Canadian Society for Epidemiology and Biostatistics (CSEB)

Video Lesson Slides

Target parameters

Balance

Propensity score matching

Propensity score matching in complex survey

Propensity score weighting in complex survey

Causal Assumptions

FAQ

Links

Target parameters

- Google Slides
- PDF Slides

Balance

- Google Slides
- PDF Slides

Propensity score matching

- Google Slides
- PDF Slides

Propensity score matching in complex survey

- Google Slides
- PDF Slides

Propensity score weighting in complex survey

- Google Slides
- PDF Slides

Causal Assumptions

- Google Slides
- PDF Slides

References

Exact Matching (CCHS)

In the following code chunk, we load the necessary R libraries for our analysis. `MatchIt` is used for matching methods to find comparable control units, `tableone` for creating Table 1 to describe baseline characteristics, `Publish` for generating readable output of regression analysis, and `survey` for analyzing complex survey samples.

```
# Load required packages
library(MatchIt)
require(tableone)
require(Publish)
require(survey)
```

Load data

In the following code chunk, we load the CCHS dataset which is related to the Canadian Community Health Survey (CCHS). We then use `ls()` to list all objects in the workspace and `str` to display the structure of the data frame, providing a quick overview of the data and checking for any character variables.

```
load("Data/propensityscore/cchs123b.RData")
ls()
#> [1] "analytic.miss"   "analytic2"       "has_annotations"
str(analytic.miss) # is there any character variable?
#> 'data.frame':    397173 obs. of  22 variables:
#>  $ CVD      : chr  "event" "no event" "no event" "no event" ...
#>  $ age      : chr  "65 years and over" "65 years and over" "30-39 years" "65 years and over
#>  $ sex      : chr  "Female" "Female" "Male" "Female" ...
#>  $ married  : chr  "single" "single" "not single" "single" ...
#>  $ race     : chr  "White" "White" "White" "White" ...
#>  $ edu      : chr  "2nd grad." "Post-2nd grad." "Post-2nd grad." "Post-2nd grad." ...
```

```
#>  $ income   : chr  "$29,999 or less" "$29,999 or less" "$80,000 or more" "$29,999 or less"
#>  $ bmi      : Factor w/ 3 levels "Underweight",..: NA NA 2 NA 2 NA 3 NA 2 3 ...
#>  $ phyact   : chr  "Inactive" "Inactive" "Inactive" "Inactive" ...
#>  $ doctor   : chr  "Yes" "Yes" "Yes" "Yes" ...
#>  $ stress   : chr  "Not too stressed" "Not too stressed" "stressed" "Not too stressed" ...
#>  $ smoke    : Factor w/ 3 levels "Never smoker",..: 3 1 3 3 2 2 3 1 2 2 ...
#>  $ drink    : Factor w/ 3 levels "Never drank",..: 2 1 2 2 2 2 3 1 2 2 ...
#>  $ fruit    : Factor w/ 3 levels "0-3 daily serving",..: 2 2 2 3 3 3 2 2 2 2 ...
#>  $ bp       : chr  "No" "No" "No" "No" ...
#>  $ diab     : chr  "No" "No" "No" "No" ...
#>  $ province : Factor w/ 2 levels "South","North": 1 1 1 1 1 1 1 1 1 1 ...
#>  $ weight   : num  142.8 71.4 168.3 71.4 196.1 ...
#>  $ cycle    : Factor w/ 3 levels "11","21","31": 1 1 1 1 1 1 1 1 1 1 ...
#>  $ ID       : int  1 2 3 4 5 6 7 8 9 10 ...
#>  $ OA       : chr  "Control" "Control" "Control" "Control" ...
#>  $ immigrate: Factor w/ 3 levels "not immigrant",..: 1 1 3 1 1 1 1 1 1 1 ...
```

Data pre-processing

In the following code chunk, we define a vector containing the names of variables of interest that needs to be converted to factor variables. We then convert these variables to factors, ensuring they are treated as categorical in subsequent analyses. We also `recode` the Osteoarthritis (OA) variable into a numeric binary format and display the frequency table of OA before and after the transformation.

```r
var.names <- c("age", "sex", "stress", "married", "income", "race",
               "bmi", "phyact", "smoke", "doctor", "drink", "bp", "province",
               "immigrate", "fruit", "diab", "edu", "CVD", "OA")
analytic.miss[var.names] <- lapply(analytic.miss[var.names] , factor)
table(analytic.miss$OA)
#>
#> Control      OA
#>  314542   40943
analytic.miss$OA <- as.numeric(analytic.miss$OA=="OA")
table(analytic.miss$OA)
#>
```

```
#>      0      1
#> 314542  40943
```

Identify subjects with missing

In the following code chunk, we create a new variable `miss` and initially assign all its values to 1 in the full dataset (that contains some missing observations). We then adjust this assignment by setting miss to 0 for observations that are also present in another complete case dataset. That means any row with `miss` equal to 0 means that row has no missing observations. Finally, we display the frequency table of the miss variable to check the number of missing and non-missing observations.

```
analytic.miss$miss <- 1
head(analytic.miss$ID) # full data
#> [1] 1 2 3 4 5 6
head(analytic2$ID) # complete case
#> [1]  3  5  7 10 11 13
head(analytic.miss$ID[analytic.miss$ID %in% analytic2$ID])
#> [1]  3  5  7 10 11 13
# if associated with complete case, assign miss <- 0
analytic.miss$miss[analytic.miss$ID %in% analytic2$ID] <- 0
table(analytic.miss$miss)
#>
#>      0      1
#> 185613 211560
```

Setting Design

Unconditional design

In the following code chunk, we explore the summary of the weight variable and establish an unconditional survey design object `w.design0` using the `svydesign` function, which will be used for subsequent survey-weighted analyses. We then explore the summary, standard deviation, and sum of the weights within our design object.

```r
summary(analytic.miss$weight)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    1.17   65.28  126.63  200.09  243.21 7154.95
w.design0 <- svydesign(id=~1, weights=~weight,
                       data=analytic.miss)
summary(weights(w.design0))
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    1.17   65.28  126.63  200.09  243.21 7154.95
sd(weights(w.design0))
#> [1] 241.0279
sum(weights(w.design0))
#> [1] 79468929
```

Conditioning the design

In the following code chunk, we create a new survey design object `w.design` by subsetting `w.design0` to only include observations without missing data `(miss == 0)`. We then explore the summary, standard deviation, and sum of the weights within this new design object.

```r
w.design <- subset(w.design0, miss == 0)
summary(weights(w.design))
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    1.17   71.56  137.95  214.61  261.91 7154.95
sd(weights(w.design))
#> [1] 254.9346
sum(weights(w.design))
#> [1] 39835061
```

Subset data (more!)

We subset the data for fast results (less computation). We will only work with cycle 1.1, and the people from Northern provinces in Canada.

```r
w.design1 <- subset(w.design, cycle == 11 & province == "North")
sum(weights(w.design1))
#> [1] 42786.28
```

Preliminary analysis

Table 1

In the following code chunk, we define a new variable vector `var.names` and create a categorical table using `svyCreateCatTable` to explore the distribution of `age` and `sex` across strata of `OA` within our subsetted design object `w.design1`. We then print the table with standardized mean differences (SMD) to assess the balance of these variables across strata.

```
var.names <- c("age", "sex")
tab0 <- svyCreateCatTable(var = var.names, strata= "OA", data=w.design1,test=FALSE)
print(tab0, smd = TRUE)
#>                        Stratified by OA
#>                         0                1              SMD
#>   n                     40691.2          2095.1
#>   age (%)                                                  1.084
#>      20-29 years        10889.4 (26.8)    120.9 ( 5.8)
#>      30-39 years        12251.7 (30.1)    237.8 (11.3)
#>      40-49 years        11094.0 (27.3)    572.7 (27.3)
#>      50-59 years         5346.6 (13.1)   1092.4 (52.1)
#>      60-64 years         1109.4 ( 2.7)     71.4 ( 3.4)
#>      65 years and over      0.0 ( 0.0)      0.0 ( 0.0)
#>      teen                   0.0 ( 0.0)      0.0 ( 0.0)
#>   sex = Male (%)        20824.6 (51.2)   1050.8 (50.2)   0.020
```

Treatment effect

In the following code chunk, we fit a logistic regression model using `svyglm` to estimate the effect of `OA` and other covariates on the binary outcome `CVD` (cardiovascular disease). We then use `publish` to display the results in a readable format.

```
fit.outcome <- svyglm(I(CVD=="event") ~ OA + age + sex + stress + married +
                        income + race + bmi + phyact + smoke +
                        immigrate + fruit + diab + edu,
                    design = w.design1,
```

```r
                    family = binomial(logit))
#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
publish(fit.outcome)
#>   Variable             Units OddsRatio         CI.95    p-value
#>         OA                          0.89   [0.17;4.59]   0.887411
#>        age       20-29 years       Ref
#>                  30-39 years      2.62  [0.29;23.43]   0.389521
#>                  40-49 years      4.89  [0.59;40.73]   0.142280
#>                  50-59 years     17.95 [2.59;124.68]   0.003550
#>                  60-64 years     23.95 [3.41;168.27]   0.001439
#>        sex            Female       Ref
#>                         Male      1.32   [0.64;2.71]   0.456222
#>     stress  Not too stressed       Ref
#>                     stressed      0.54   [0.21;1.39]   0.198815
#>    married        not single       Ref
#>                       single      0.75   [0.31;1.80]   0.513807
#>     income   $29,999 or less       Ref
#>             $30,000-$49,999      0.72   [0.24;2.16]   0.556703
#>             $50,000-$79,999      0.95   [0.27;3.40]   0.939104
#>             $80,000 or more      0.47   [0.10;2.15]   0.332557
#>       race         Non-white       Ref
#>                        White      0.33   [0.11;0.94]   0.038131
#>        bmi       Underweight       Ref
#>              healthy weight      0.29   [0.03;3.20]   0.310237
#>                  Overweight      0.44   [0.04;4.77]   0.503130
#>     phyact            Active       Ref
#>                     Inactive      0.84   [0.30;2.40]   0.751345
#>                     Moderate      1.02   [0.32;3.27]   0.979528
#>      smoke      Never smoker       Ref
#>              Current smoker      0.98   [0.26;3.76]   0.981454
#>               Former smoker      0.71   [0.18;2.71]   0.612518
#>  immigrate     not immigrant       Ref
#>                  > 10 years      0.14   [0.03;0.78]   0.025010
#>                       recent      0.00   [0.00;0.00]    < 1e-04
#>      fruit 0-3 daily serving       Ref
#>            4-6 daily serving      1.15   [0.52;2.56]   0.725722
#>             6+ daily serving      0.68   [0.17;2.71]   0.583752
#>       diab                 No       Ref
#>                           Yes      3.08  [0.93;10.23]   0.066677
#>        edu          < 2ndary       Ref
```

```
#>                    2nd grad.      4.12  [0.87;19.43]   0.074178
#>               Other 2nd grad.      3.04  [0.63;14.67]   0.167135
#>                Post-2nd grad.      3.00  [0.82;10.98]   0.096939
```

Matching: Estimating treatment effect

Going back to the data (**not working on design here while matching**)

In the following code chunk, we create a new dataset by omitting `NA` values from `analytic.miss` and converting it to a data frame. We then create a subset `analytic11n` which includes only observations from cycle 1.1 and the Northern provinces. We display the dimensions of this subset, as well as frequency tables of `OA` and a cross-tabulation of `OA` and `age` to understand the distribution of our target variable and a key covariate.

```r
# Create the dataset without design features
analytic2 <- as.data.frame(na.omit(analytic.miss))
analytic11n <- subset(analytic2, cycle == 11 & province == "North")
dim(analytic11n)
#> [1] 1424   23
table(analytic11n$OA)
#>
#>    0    1
#> 1357   67
table(analytic11n$OA,analytic11n$age)
#>
#>     20-29 years 30-39 years 40-49 years 50-59 years 60-64 years
#>   0         345         432         358         177          45
#>   1           4          11          18          31           3
#>
#>     65 years and over teen
#>   0                  0    0
#>   1                  0    0
```

Matching by 1 matching variable

In the following code chunk, we perform exact matching using a single variable, `age`. We define the matching formula and apply the `matchit` function to create matched sets of treated and control units. The resulting `matching.obj` object is displayed to summarize the matching results.

```
match.formula <- as.formula("OA ~ age")
matching.obj <- matchit(match.formula,
                        data = analytic11n,
                        method = "exact")
matching.obj
#> A matchit object
#>  - method: Exact matching
#>  - number of obs.: 1424 (original), 1424 (matched)
#>  - target estimand: ATT
#>  - covariates: age
```

Matching by 2 matching variables

In the following code chunk, we extend the matching to include two variables, `age` and `sex`. We create a new variable `var.comb` that concatenates these two variables and display its frequency table and the number of unique combinations. We then perform exact matching using both variables and display the resulting object.

```
var.comb <- do.call('paste0',
                    analytic11n[, c('age', 'sex')])
table(var.comb)
#> var.comb
#> 20-29 yearsFemale   20-29 yearsMale 30-39 yearsFemale   30-39 yearsMale
#>               184               165               220               223
#> 40-49 yearsFemale   40-49 yearsMale 50-59 yearsFemale   50-59 yearsMale
#>               187               189               101               107
#> 60-64 yearsFemale   60-64 yearsMale
#>                24                24
length(table(var.comb))
#> [1] 10
```

```r
match.formula <- as.formula("OA ~ age + sex")
matching.obj <- matchit(match.formula,
                        data = analytic11n,
                        method = "exact")
matching.obj
#> A matchit object
#>  - method: Exact matching
#>  - number of obs.: 1424 (original), 1424 (matched)
#>  - target estimand: ATT
#>  - covariates: age, sex
```

Matching by 3 matching variables

In the following code chunk, we further extend the matching to include three variables: `age`, `sex`, and `stress`. We explore the unique combinations of these variables and their distribution across levels of `OA`. We then perform exact matching using these three variables and display the resulting object.

```r
var.comb <- do.call('paste0',
                    analytic11n[, c('age', 'sex', 'stress')])
table(var.comb)
#> var.comb
#> 20-29 yearsFemaleNot too stressed         20-29 yearsFemalestressed
#>                               157                                27
#>   20-29 yearsMaleNot too stressed           20-29 yearsMalestressed
#>                               147                                18
#> 30-39 yearsFemaleNot too stressed         30-39 yearsFemalestressed
#>                               170                                50
#>   30-39 yearsMaleNot too stressed           30-39 yearsMalestressed
#>                               183                                40
#> 40-49 yearsFemaleNot too stressed         40-49 yearsFemalestressed
#>                               142                                45
#>   40-49 yearsMaleNot too stressed           40-49 yearsMalestressed
#>                               141                                48
#> 50-59 yearsFemaleNot too stressed         50-59 yearsFemalestressed
#>                                72                                29
#>   50-59 yearsMaleNot too stressed           50-59 yearsMalestressed
#>                                78                                29
```

```
#> 60-64 yearsFemaleNot too stressed         60-64 yearsFemalestressed
#>                                 18                                 6
#>   60-64 yearsMaleNot too stressed           60-64 yearsMalestressed
#>                                 20                                 4
length(table(var.comb))
#> [1] 20
table(var.comb,analytic11n$OA)
#>
#> var.comb                              0   1
#>   20-29 yearsFemaleNot too stressed 156   1
#>   20-29 yearsFemalestressed          27   0
#>   20-29 yearsMaleNot too stressed   144   3
#>   20-29 yearsMalestressed            18   0
#>   30-39 yearsFemaleNot too stressed 168   2
#>   30-39 yearsFemalestressed          49   1
#>   30-39 yearsMaleNot too stressed   178   5
#>   30-39 yearsMalestressed            37   3
#>   40-49 yearsFemaleNot too stressed 130  12
#>   40-49 yearsFemalestressed          42   3
#>   40-49 yearsMaleNot too stressed   138   3
#>   40-49 yearsMalestressed            48   0
#>   50-59 yearsFemaleNot too stressed  65   7
#>   50-59 yearsFemalestressed          22   7
#>   50-59 yearsMaleNot too stressed    67  11
#>   50-59 yearsMalestressed            23   6
#>   60-64 yearsFemaleNot too stressed  17   1
#>   60-64 yearsFemalestressed           5   1
#>   60-64 yearsMaleNot too stressed    19   1
#>   60-64 yearsMalestressed             4   0
match.formula <- as.formula("OA ~ age + sex + stress")
matching.obj <- matchit(match.formula,
                        data = analytic11n,
                        method = "exact")
matching.obj
#> A matchit object
#>  - method: Exact matching
#>  - number of obs.: 1424 (original), 1327 (matched)
#>  - target estimand: ATT
#>  - covariates: age, sex, stress
```

Matching by 4 matching variables

The process of matching by 4 variables involves creating combinations of the 4 variables, exploring their distributions, and performing exact matching.

```
var.comb <- do.call('paste0',
                    analytic11n[, c('age', 'sex',
                                    'stress','income')])
#table(var.comb)
length(table(var.comb))
#> [1] 76
match.formula <- as.formula("OA ~ age + sex + stress + income")
matching.obj <- matchit(match.formula,
                        data = analytic11n,
                        method = "exact")
matching.obj
#> A matchit object
#>  - method: Exact matching
#>  - number of obs.: 1424 (original), 900 (matched)
#>  - target estimand: ATT
#>  - covariates: age, sex, stress, income
```

Matching by 5 matching variables

```
var.comb <- do.call('paste0',
                    analytic11n[, c('age', 'sex',
                                    'stress','income','race')])
length(table(var.comb))
#> [1] 146
match.formula <- as.formula("OA ~ age + sex + stress + income + race")
matching.obj <- matchit(match.formula,
                        data = analytic11n,
                        method = "exact")
matching.obj
#> A matchit object
#>  - method: Exact matching
#>  - number of obs.: 1424 (original), 616 (matched)
```

```
#>  - target estimand: ATT
#>  - covariates: age, sex, stress, income, race
```

Matching by 6 matching variables

```
var.comb <- do.call('paste0',
                    analytic11n[, c('age', 'sex',
                                    'stress','income','race','edu')])
length(table(var.comb))
#> [1] 354
match.formula <- as.formula("OA ~ age + sex + stress + income + race + edu")
matching.obj <- matchit(match.formula,
                        data = analytic11n,
                        method = "exact")
matching.obj
#> A matchit object
#>  - method: Exact matching
#>  - number of obs.: 1424 (original), 399 (matched)
#>  - target estimand: ATT
#>  - covariates: age, sex, stress, income, race, edu
OACVD.match.11n <- match.data(matching.obj)
var.names <- c("age", "sex", "stress", "income", "race", "edu")
tab1 <- CreateCatTable(var = var.names, strata= "OA", data=OACVD.match.11n,test=FALSE)
print(tab1, smd = TRUE)
#>                        Stratified by OA
#>                         0           1          SMD
#>   n                     337         62
#>   age (%)                                       0.565
#>      20-29 years         63 (18.7)   4 ( 6.5)
#>      30-39 years         61 (18.1)  11 (17.7)
#>      40-49 years        127 (37.7)  17 (27.4)
#>      50-59 years         82 (24.3)  28 (45.2)
#>      60-64 years          4 ( 1.2)   2 ( 3.2)
#>      65 years and over    0 ( 0.0)   0 ( 0.0)
#>      teen                 0 ( 0.0)   0 ( 0.0)
#>   sex = Male (%)        211 (62.6)  29 (46.8)   0.322
#>   stress = stressed (%)  42 (12.5)  17 (27.4)   0.381
#>   income (%)                                    0.115
```

```
#>      $29,999 or less     69 (20.5)  11 (17.7)
#>      $30,000-$49,999     57 (16.9)  13 (21.0)
#>      $50,000-$79,999     69 (20.5)  12 (19.4)
#>      $80,000 or more    142 (42.1)  26 (41.9)
#>   race = White (%)      242 (71.8)  43 (69.4)   0.054
#>   edu (%)                                       0.146
#>      < 2ndary            73 (21.7)  11 (17.7)
#>      2nd grad.            5 ( 1.5)   2 ( 3.2)
#>      Other 2nd grad.      0 ( 0.0)   0 ( 0.0)
#>      Post-2nd grad.     259 (76.9)  49 (79.0)
```

Treatment effect

Convert data to design

In the following code chunk, we create a new variable matched in the `analytic.miss` dataset to indicate whether an observation was included in the matched dataset `OACVD.match.11n`. We then create a new survey design object `w.design.m` that includes only the matched observations for subsequent analyses.

```
analytic.miss$matched <- 0
length(analytic.miss$ID) # full data
#> [1] 397173
length(OACVD.match.11n$ID) # matched data
#> [1] 399
length(analytic.miss$ID[analytic.miss$ID %in% OACVD.match.11n$ID])
#> [1] 399
analytic.miss$matched[analytic.miss$ID %in% OACVD.match.11n$ID] <- 1
table(analytic.miss$matched)
#>
#>      0      1
#> 396774    399
w.design0 <- svydesign(id=~1, weights=~weight,
                       data=analytic.miss)
w.design.m <- subset(w.design0, matched == 1)
```

Outcome analysis

The subsequent code chunks involve fitting logistic regression models to estimate the treatment effect, both in a crude and adjusted manner, respectively. The models are fitted using the matched survey design object and the results are displayed in a readable format.

0.0.0.0.1 * Crude

```
fit.outcome <- svyglm(I(CVD=="event") ~ OA,
                   design = w.design.m,
                   family = binomial(logit))
#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
publish(fit.outcome)
#>  Variable Units OddsRatio        CI.95  p-value
#>        OA            3.14 [0.80;12.40]   0.1025
```

0.0.0.0.2 * Adjusted

```
fit.outcome <- svyglm(I(CVD=="event") ~ OA +
                        age + sex + stress + income + race + edu,
                   design = w.design.m,
                   family = binomial(logit))
#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
#> Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
publish(fit.outcome)
#>  Variable            Units    OddsRatio                        CI.95   p-value
#>        OA                         2.04                  [0.34;12.16]   0.43593
#>       age      20-29 years          Ref
#>                30-39 years         0.54                   [0.08;3.51]   0.51962
#>                40-49 years  30148597.83    [7758796.64;117149345.85]   < 1e-04
#>                50-59 years  63290825.92   [12589874.04;318170669.03]   < 1e-04
#>                60-64 years         0.31                   [0.01;9.33]   0.49735
#>       sex           Female          Ref
#>                       Male         1.58                   [0.29;8.62]   0.59729
#>    stress Not too stressed          Ref
#>                   stressed         0.15                   [0.01;1.80]   0.13666
#>    income  $29,999 or less          Ref
```

```
#>            $30,000-$49,999         0.20                    [0.01;3.84]   0.28640
#>            $50,000-$79,999         0.20                    [0.02;1.95]   0.16543
#>            $80,000 or more         0.08                    [0.01;0.68]   0.02122
#>      race        Non-white          Ref
#>                    White         1.02                    [0.11;9.45]   0.98723
#>       edu         < 2ndary          Ref
#>                  2nd grad. 845233463.16 [44642866.24;16002995941.97]   < 1e-04
#>             Post-2nd grad.  69867459.11    [9660580.05;505296971.72]   < 1e-04
```

Questions for the students

- Look at all the ORs. Some of them are VERY high. Why?
- Look at the CI in the above table. Some of them are Inf. Why?
- Should we match matching variables in the regression?

Matching by a lot of variables

The code chunks involve performing matching using a large number of variables and estimating the treatment effect using the matched data. The process involves creating matched datasets, converting them to survey design objects, and fitting logistic regression models.

Matching part in data

```r
match.formula <- as.formula("OA ~ age + sex + stress + married +
                         income + race + bmi + phyact + smoke +
                         immigrate + fruit + diab + edu")
matching.obj2 <- matchit(match.formula,
                        data = analytic11n,
                        method = "exact")
matching.obj2
#> A matchit object
#>  - method: Exact matching
#>  - number of obs.: 1424 (original), 22 (matched)
#>  - target estimand: ATT
```

```
#>  - covariates: age, sex, stress, married, income, race, bmi, phyact, smoke, immigrate, frui
OACVD.match.11n2 <- match.data(matching.obj2)
var.names <- c("age", "sex", "stress", "married", "income", "race",
               "bmi", "phyact", "smoke", "immigrate", "fruit", "diab", "edu")
tab2 <- CreateCatTable(var = var.names, strata= "OA", data=OACVD.match.11n2,test=FALSE)
print(tab2, smd = TRUE)
#>                          Stratified by OA
#>                           0          1          SMD
#>   n                       11         11
#>   age (%)                                       <0.001
#>      20-29 years           3 (27.3)   3 (27.3)
#>      30-39 years           1 ( 9.1)   1 ( 9.1)
#>      40-49 years           4 (36.4)   4 (36.4)
#>      50-59 years           3 (27.3)   3 (27.3)
#>      60-64 years           0 ( 0.0)   0 ( 0.0)
#>      65 years and over     0 ( 0.0)   0 ( 0.0)
#>      teen                  0 ( 0.0)   0 ( 0.0)
#>   sex = Male (%)           6 (54.5)   6 (54.5)  <0.001
#>   stress = stressed (%)    1 ( 9.1)   1 ( 9.1)  <0.001
#>   married = single (%)     3 (27.3)   3 (27.3)  <0.001
#>   income (%)                                    <0.001
#>      $29,999 or less       1 ( 9.1)   1 ( 9.1)
#>      $30,000-$49,999       2 (18.2)   2 (18.2)
#>      $50,000-$79,999       2 (18.2)   2 (18.2)
#>      $80,000 or more       6 (54.5)   6 (54.5)
#>   race = White (%)        10 (90.9)  10 (90.9)  <0.001
#>   bmi (%)                                       <0.001
#>      Underweight           0 ( 0.0)   0 ( 0.0)
#>      healthy weight        4 (36.4)   4 (36.4)
#>      Overweight            7 (63.6)   7 (63.6)
#>   phyact (%)                                    <0.001
#>      Active                3 (27.3)   3 (27.3)
#>      Inactive              5 (45.5)   5 (45.5)
#>      Moderate              3 (27.3)   3 (27.3)
#>   smoke (%)                                     <0.001
#>      Never smoker          3 (27.3)   3 (27.3)
#>      Current smoker        2 (18.2)   2 (18.2)
#>      Former smoker         6 (54.5)   6 (54.5)
#>   immigrate (%)                                 <0.001
#>      not immigrant        10 (90.9)  10 (90.9)
```

```
#>      > 10 years          1 ( 9.1)   1 ( 9.1)
#>      recent              0 ( 0.0)   0 ( 0.0)
#>   fruit (%)                                     <0.001
#>      0-3 daily serving   3 (27.3)   3 (27.3)
#>      4-6 daily serving   6 (54.5)   6 (54.5)
#>      6+ daily serving    2 (18.2)   2 (18.2)
#>   diab = Yes (%)         0 ( 0.0)   0 ( 0.0)  <0.001
#>   edu (%)                                       <0.001
#>      < 2ndary            1 ( 9.1)   1 ( 9.1)
#>      2nd grad.           0 ( 0.0)   0 ( 0.0)
#>      Other 2nd grad.     0 ( 0.0)   0 ( 0.0)
#>      Post-2nd grad.     10 (90.9)  10 (90.9)
```

Treatment effect estimation in design

0.0.0.0.1 * Create design

```
analytic.miss$matched2 <- 0
length(analytic.miss$ID) # full data
#> [1] 397173
length(OACVD.match.11n2$ID) # matched data
#> [1] 22
length(analytic.miss$ID[analytic.miss$ID %in% OACVD.match.11n2$ID])
#> [1] 22
analytic.miss$matched2[analytic.miss$ID %in% OACVD.match.11n2$ID] <- 1
table(analytic.miss$matched2)
#>
#>      0      1
#> 397151     22
w.design0 <- svydesign(id=~1, weights=~weight,
                       data=analytic.miss)
w.design.m2 <- subset(w.design0, matched2 == 1)
```

0.0.0.0.2 * outcome analysis

```
fit.outcome <- svyglm(I(CVD=="event") ~ OA + age + sex + stress + married +
                        income + race + bmi + phyact + smoke +
                        immigrate + fruit + diab + edu,
```

```
                    design = w.design.m2,
                    family = binomial(logit))
publish(fit.outcome)
# Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]) :
#   contrasts can be applied only to factors with 2 or more levels
```

0.0.0.0.3 * Questions for the students

- Why the above model not fitting?

Save data for later use

```
save(analytic11n, analytic2, analytic.miss, file="Data/propensityscore/cchs123c.RData")
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

PSM in OA-CVD (CCHS)

This tutorial is a comprehensive guide on implementing Propensity Score Matching (PSM) using R, particularly focusing on a OA - CVD health study from the Canadian Community Health Survey (CCHS). This PSM method is used to reduce bias due to confounding variables in observational studies by matching treated and control units with similar propensity scores. The tutorial illustrates that PSM is an iterative process, where researchers may need to refine their matching strategy to achieve satisfactory balance in the matched sample. Different strategies for estimating the treatment effect in the matched sample are explored, each with its own assumptions and implications.

Load packages

At first, various R packages are loaded to utilize their functions for data manipulation, statistical analysis, and visualization.

```r
# Load required packages
library(MatchIt)
require(tableone)
require(survey)
require(cobalt)
require(Publish)
require(optmatch)
```

Load data

The dataset is loaded into the R environment. Variables are renamed to avoid conflicts in subsequent analyses.

```r
load(file="Data/propensityscore/cchs123c.RData")
head(analytic11n)
#>            CVD         age    sex    married      race              edu
#> 17838 no event 30-39 years Female     single Non-white Other 2nd grad.
#> 17839 no event 30-39 years Female     single     White  Post-2nd grad.
#> 17842    event 50-59 years Female not single Non-white        < 2ndary
#> 17844 no event 30-39 years Female not single Non-white  Post-2nd grad.
#> 17846 no event 30-39 years   Male     single     White  Post-2nd grad.
#> 17847 no event 30-39 years   Male     single     White  Post-2nd grad.
#>                income            bmi   phyact doctor           stress
#> 17838 $50,000-$79,999 healthy weight Inactive    Yes Not too stressed
#> 17839 $50,000-$79,999 healthy weight Moderate     No Not too stressed
#> 17842 $80,000 or more healthy weight Inactive     No Not too stressed
#> 17844 $80,000 or more healthy weight Inactive    Yes Not too stressed
#> 17846 $30,000-$49,999     Overweight Inactive     No Not too stressed
#> 17847 $50,000-$79,999     Overweight Inactive    Yes Not too stressed
#>                smoke           drink             fruit  bp diab province weight
#> 17838 Current smoker Current drinker 0-3 daily serving  No   No    North  20.66
#> 17839 Current smoker Current drinker 4-6 daily serving  No   No    North  15.06
#> 17842   Never smoker     Never drank  6+ daily serving Yes   No    North  74.04
#> 17844  Former smoker Current drinker 4-6 daily serving  No   No    North  30.47
#> 17846   Never smoker Current drinker 0-3 daily serving  No   No    North  12.10
#> 17847 Current smoker Current drinker  6+ daily serving  No   No    North  16.90
#>       cycle    ID OA     immigrate miss
#> 17838    11 17838  0 not immigrant    0
#> 17839    11 17839  0 not immigrant    0
#> 17842    11 17842  0 not immigrant    0
#> 17844    11 17844  0 not immigrant    0
#> 17846    11 17846  0 not immigrant    0
#> 17847    11 17847  0 not immigrant    0

# later we will create another variable called weights
# hence to avoid any conflict/ambiguity,
# renaming weight variable to survey.weight
analytic.miss$survey.weight <- analytic.miss$weight
analytic11n$survey.weight <- analytic11n$weight
analytic.miss$weight <- analytic11n$weight <- NULL
```

Analysis

We are going to apply propensity score analysis (Matching) in our OA - CVD problem from CCHS. For computation considerations, we will only work with cycle 1.1, and the people from Northern provinces in Canada (`analytic11n` data).

Step 1

Specify PS

A logistic regression model formula is specified to calculate the propensity scores (PS), which is the probability of receiving the treatment given the observed covariates.

```
ps.formula <- as.formula("OA ~ age + sex + stress + married +
                          income + race + bmi + phyact + smoke +
                         doctor + drink + bp +
                          immigrate + fruit + diab + edu")
var.names <- c("age", "sex", "stress", "married",
               "income", "race", "bmi", "phyact", "smoke",
               "doctor", "drink", "bp",
               "immigrate", "fruit", "diab", "edu")
```

Fit model

The software fits the PS model using a logistic regression by default. This package actually performs step 1 and 2 with one command `matchit`.

Look at the website for arguments of matchit (RDocumentation 2023)]. It looks like this

```
matchit(formula, data, model="logit", discard=0, reestimate=FALSE, nearest=TRUE,
                 replace=FALSE, m.order=2, ratio=1, caliper=0, calclosest=FALSE,
                 subclass=0, sub.by="treat", mahvars=NULL, exact=FALSE, counter=TRUE, full=FALS
```

Tip

Nearest-Neighbor Matching:
Nearest-neighbor matching is a widely used technique in PSM to pair treated and control units based on the proximity of their propensity scores. It is straightforward and computationally efficient, making it a popular choice in many applications of PSM. Nearest-neighbor matching is often termed a "greedy" algorithm because it matches units in order, without considering the global distribution of propensity scores. Once a match is made, it is not revisited, even if a later unit would have been a better match. The method seeks to minimize bias by creating closely matched pairs but can increase variance if the pool of potential matches is reduced too much (e.g., using a very narrow caliper). It is essential to ensure that there is a common support region where the distributions of propensity scores for treated and control units overlap, ensuring comparability.

Step 2

Match subjects by PS

We are going to match using a Nearest neighbor algorithm. This is a greedy matching algorithm. Note that we are not even defining any caliper.

Tip

Caliper:
In the context of PSM, a caliper is a predefined maximum allowable difference between the propensity scores of matched units. Essentially, it sets a threshold for how dissimilar matched units can be in terms of their propensity scores. When a caliper is used, a treated unit is only matched with a control unit if the absolute difference in their propensity scores is less than or equal to the specified caliper width. The caliper is used to avoid bad matches

and thereby minimize bias in the estimated treatment effect. The size of the caliper is crucial. Too wide a caliper may allow poor matches, while too narrow a caliper may result in many units going unmatched. Implementing a caliper involves a trade-off between bias and efficiency. Using a caliper reduces bias by avoiding poor matches but may increase variance by reducing the number of matched pairs available for analysis. Therefore, the use of a caliper in PSM is a strategic decision to enhance the quality of matches and thereby improve the validity of causal inferences drawn from observational data. It is a practical tool to ensure that matched units are sufficiently similar in terms of their propensity scores, reducing the likelihood of bias due to poor matches.

```r
# set seed
set.seed(123)
# match
matching.obj <- matchit(ps.formula,
                        data = analytic11n,
                        method = "nearest",
                        ratio = 1)
# see how many matched
matching.obj
#> A matchit object
#>  - method: 1:1 nearest neighbor matching without replacement
#>  - distance: Propensity score
#>              - estimated with logistic regression
#>  - number of obs.: 1424 (original), 134 (matched)
#>  - target estimand: ATT
#>  - covariates: age, sex, stress, married, income, race, bmi, phyact, smoke, doctor, drink,
# create the "matched"" data
OACVD.match <- match.data(matching.obj)
# see the dimension
dim(analytic11n)
#> [1] 1424   23
dim(OACVD.match)
#> [1] 134  26
```

Let's try to understand how this is working.

Extract matched IDs

```
m.mat<-matching.obj$match.matrix
head(m.mat)
#>       [,1]
#> 17864 "96719"
#> 17921 "17846"
#> 18191 "97168"
#> 18256 "111999"
#> 18264 "17989"
#> 18383 "108197"
```

Extract the matched treated IDs

```
treated.id<-as.numeric(row.names(m.mat))
treated.id # basically row names
#>  [1]  17864  17921  18191  18256  18264  18383  18389  18475  39105  96344
#> [11]  96364  96407  96424  96460  96484  96571  96582  96625  96632  96641
#> [21]  96657  96686  96693  96696  96705  96734  96795  96840  96913  97027
#> [31]  97065  97125 108178 108183 108185 108192 111809 111813 111856 111859
#> [41] 111895 111896 111920 111942 112014 112026 112046 112083 112086 112114
#> [51] 112122 112151 112167 112189 112197 112215 112232 112245 112275 112284
#> [61] 112289 112290 112300 112325 112375 126477 126522
```

Extract the matched untreated IDs

```
untreated.id <- as.numeric(m.mat)
untreated.id # basically row names
#>  [1]  96719  17846  97168 111999  17989 108197 112384  17909 126561 111880
#> [11] 112184  18117  96865  18120  97023 112379  97017 126562  96356 126470
#> [21] 126385  96374  18203  18262  96972 111924  96354  96983  18235  96882
#> [31] 112054  18321  18426 112349  38996 126516 111814 112087  96569 111932
#> [41] 126539  18315  96665  18225 112052 112324 112165  18329  96609 126376
#> [51]  96474 126570 126547 126343  96680  96558 111931  96718  96533 111823
#> [61] 112177  17953  17904 111908 111962  96644  96576
```

Extract the matched treated data

```
tx <- analytic11n[rownames(analytic11n) %in% treated.id,]
head(tx[c("OA", "CVD", "sex", "age", "race", "edu")])
#>       OA      CVD    sex         age      race            edu
#> 17864  1 no event Female 40-49 years Non-white       < 2ndary
#> 17921  1 no event Female 30-39 years     White Post-2nd grad.
#> 18191  1 no event Female 40-49 years Non-white       < 2ndary
#> 18256  1 no event   Male 30-39 years Non-white Post-2nd grad.
#> 18264  1 no event   Male 20-29 years Non-white       < 2ndary
#> 18383  1 no event   Male 30-39 years Non-white       < 2ndary
```

Extract the matched untreated data

```
utx <- analytic11n[rownames(analytic11n) %in% untreated.id,]
head(utx[c("OA", "CVD", "sex", "age", "race", "edu")])
#>       OA      CVD    sex         age      race            edu
#> 17846  0 no event   Male 30-39 years     White Post-2nd grad.
#> 17904  0 no event Female 40-49 years Non-white       < 2ndary
#> 17909  0 no event Female 20-29 years     White Post-2nd grad.
#> 17953  0 no event   Male 20-29 years     White Post-2nd grad.
#> 17989  0    event   Male 30-39 years Non-white      2nd grad.
#> 18117  0 no event Female 60-64 years Non-white       < 2ndary
```

Extract the matched data altogether

Simply using `match.data` is enough (as done earlier).

```
OACVD.match <- match.data(matching.obj)
```

Assign match ID

```r
OACVD.match$match.ID <- NA
OACVD.match$match.ID[rownames(OACVD.match) %in% treated.id] <- 1:length(treated.id)
OACVD.match$match.ID[rownames(OACVD.match) %in% untreated.id] <- 1:length(untreated.id)
table(OACVD.match$match.ID)
#>
#>  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
#>  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
#> 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
#>  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
#> 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67
#>  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
```

Take a look at individual matches for the first match

```r
na.omit(OACVD.match[OACVD.match$match.ID == 1,])
#>            CVD         age    sex    married      race             edu
#> 17846 no event 30-39 years   Male     single     White Post-2nd grad.
#> 17864 no event 40-49 years Female not single Non-white       < 2ndary
#>                income        bmi   phyact doctor           stress
#> 17846 $30,000-$49,999 Overweight Inactive     No Not too stressed
#> 17864 $30,000-$49,999 Overweight Inactive    Yes         stressed
#>                smoke           drink             fruit bp diab province cycle
#> 17846   Never smoker Current drinker 0-3 daily serving No   No    North    11
#> 17864 Current smoker Current drinker 4-6 daily serving No   No    North    11
#>          ID OA     immigrate miss survey.weight   distance weights subclass
#> 17846 17846  0 not immigrant    0         12.10 0.01854076       1       46
#> 17864 17864  1 not immigrant    0         38.23 0.18190027       1       30
#>       match.ID
#> 17846        1
#> 17864        1
```

Take a look at individual matches for the second match

```r
na.omit(OACVD.match[OACVD.match$match.ID == 2,])
#>            CVD         age    sex    married      race             edu
#> 17904 no event 40-49 years Female not single Non-white       < 2ndary
#> 17921 no event 30-39 years Female     single     White Post-2nd grad.
#>                income        bmi   phyact doctor           stress
#> 17904 $30,000-$49,999 Overweight Inactive    Yes Not too stressed
```

```
#> 17921 $29,999 or less Overweight Inactive     No         stressed
#>                 smoke           drink             fruit  bp diab province cycle
#> 17904 Current smoker   Former driker 0-3 daily serving Yes   No    North    11
#> 17921   Never smoker Current drinker 0-3 daily serving  No   No    North    11
#>          ID OA     immigrate miss survey.weight   distance weights subclass
#> 17904 17904  0 not immigrant    0         52.04 0.06783228       1       25
#> 17921 17921  1 not immigrant    0         15.88 0.01857417       1       46
#>       match.ID
#> 17904        2
#> 17921        2
```

Step 3

Both graphical and numerical methods are used to assess the quality of the matches and the balance of covariates in the matched sample.

Examining PS graphically

Visually inspect the PS and assess the balance of covariates in the matched sample. Various plots are generated to visualize the distribution of PS across treatment groups and to check the balance of covariates before and after matching.

matchit package

```r
# plot(matching.obj) # covariate balance
plot(matching.obj, type = "jitter") # propensity score locations
```

```
#> [1] "To identify the units, use first mouse button; to stop, use second."
#> integer(0)
plot(matching.obj, type = "hist") #check matched treated vs matched control
```

```
summrize.output <- summary(matching.obj, standardize = TRUE)
plot(summrize.output)
```

Overalp check

```
# plot propensity scores by exposure group
plot(density(OACVD.match$distance[OACVD.match$OA==1]),
     col = "red", main = "")
lines(density(OACVD.match$distance[OACVD.match$OA==0]),
      col = "blue", lty = 2)
legend("topright", c("Non-arthritis","OA"),
       col = c("red", "blue"), lty=1:2)
```

cobalt package

Overlap check in a more convenient way

```
# different badwidth
bal.plot(matching.obj, var.name = "distance")
```

Look at the data

```
# what is distance variable here?
head(OACVD.match)
#>            CVD         age    sex    married      race            edu
#> 17846 no event 30-39 years   Male     single     White Post-2nd grad.
#> 17864 no event 40-49 years Female not single Non-white       < 2ndary
#> 17904 no event 40-49 years Female not single Non-white       < 2ndary
#> 17909 no event 20-29 years Female not single     White Post-2nd grad.
#> 17921 no event 30-39 years Female     single     White Post-2nd grad.
#> 17953 no event 20-29 years   Male not single     White Post-2nd grad.
#>                income        bmi   phyact doctor           stress
#> 17846 $30,000-$49,999 Overweight Inactive     No Not too stressed
#> 17864 $30,000-$49,999 Overweight Inactive    Yes         stressed
#> 17904 $30,000-$49,999 Overweight Inactive    Yes Not too stressed
#> 17909 $80,000 or more Overweight Inactive    Yes Not too stressed
```

```
#> 17921 $29,999 or less Overweight Inactive     No         stressed
#> 17953 $50,000-$79,999 Overweight Moderate     No Not too stressed
#>                smoke           drink             fruit  bp diab province cycle
#> 17846   Never smoker Current drinker 0-3 daily serving  No   No    North    11
#> 17864 Current smoker Current drinker 4-6 daily serving  No   No    North    11
#> 17904 Current smoker   Former driker 0-3 daily serving Yes   No    North    11
#> 17909   Never smoker Current drinker 4-6 daily serving  No   No    North    11
#> 17921   Never smoker Current drinker 0-3 daily serving  No   No    North    11
#> 17953   Never smoker Current drinker 0-3 daily serving  No   No    North    11
#>          ID OA     immigrate miss survey.weight   distance weights subclass
#> 17846 17846  0 not immigrant    0         12.10 0.01854076       1       46
#> 17864 17864  1 not immigrant    0         38.23 0.18190027       1       30
#> 17904 17904  0 not immigrant    0         52.04 0.06783228       1       25
#> 17909 17909  0 not immigrant    0         33.23 0.03126371       1       36
#> 17921 17921  1 not immigrant    0         15.88 0.01857417       1       46
#> 17953 17953  0 not immigrant    0         26.91 0.00467081       1       24
#>       match.ID
#> 17846        1
#> 17864        1
#> 17904        2
#> 17909        3
#> 17921        2
#> 17953        4
```

Numerical values of PS

```
summary(OACVD.match$distance)
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> 0.004671 0.044834 0.099094 0.138969 0.200576 0.611166
by(OACVD.match$distance, OACVD.match$OA, summary)
#> OACVD.match$OA: 0
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> 0.004671 0.047344 0.099215 0.135042 0.199279 0.418206
#> ------------------------------------------------------------
#> OACVD.match$OA: 1
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> 0.004671 0.047346 0.098973 0.142897 0.198741 0.611166
```

Question for the students

- Are you happy with the matching after reviewing the plots?

Covariate balance in matched sample

Covariate balance is assessed numerically using standardized mean differences (SMD).

> Tip
>
> **Standardized mean differences**: SMD is a versatile and widely used statistical measure that facilitates the comparison of groups in research by providing a scale-free metric of difference and balance. In the context of propensity score matching, achieving low SMD values for covariates after matching is crucial to ensuring the validity of causal inferences drawn from the matched sample.
> Benifits:
>
> - SMD is not influenced by the scale of the measured variable, making it suitable for comparing the balance of different variables measured on different scales.
> - Unlike hypothesis testing, SMD is not affected by sample size, making it a reliable measure for assessing balance in matched samples.

```r
tab1 <- CreateTableOne(strata = "OA", data = OACVD.match,
                       test = FALSE, vars = var.names)
print(tab1, smd = TRUE)
#>                          Stratified by OA
#>                           0          1          SMD
#>   n                       67         67
#>   age (%)                                        0.190
#>      20-29 years           4 ( 6.0)   4 ( 6.0)
#>      30-39 years          16 (23.9)  11 (16.4)
#>      40-49 years          16 (23.9)  18 (26.9)
#>      50-59 years          28 (41.8)  31 (46.3)
```

```
#>      60-64 years          3 ( 4.5)   3 ( 4.5)
#>      65 years and over    0 ( 0.0)   0 ( 0.0)
#>      teen                 0 ( 0.0)   0 ( 0.0)
#>   sex = Male (%)         28 (41.8)  32 (47.8)   0.120
#>   stress = stressed (%) 20 (29.9)  21 (31.3)   0.032
#>   married = single (%)  22 (32.8)  23 (34.3)   0.032
#>   income (%)                                   0.183
#>      $29,999 or less    11 (16.4)  13 (19.4)
#>      $30,000-$49,999    19 (28.4)  15 (22.4)
#>      $50,000-$79,999    14 (20.9)  12 (17.9)
#>      $80,000 or more    23 (34.3)  27 (40.3)
#>   race = White (%)      43 (64.2)  44 (65.7)   0.031
#>   bmi (%)                                     <0.001
#>      Underweight         0 ( 0.0)   0 ( 0.0)
#>      healthy weight     18 (26.9)  18 (26.9)
#>      Overweight         49 (73.1)  49 (73.1)
#>   phyact (%)                                   0.041
#>      Active             16 (23.9)  16 (23.9)
#>      Inactive           40 (59.7)  39 (58.2)
#>      Moderate           11 (16.4)  12 (17.9)
#>   smoke (%)                                    0.258
#>      Never smoker       14 (20.9)   9 (13.4)
#>      Current smoker     20 (29.9)  27 (40.3)
#>      Former smoker      33 (49.3)  31 (46.3)
#>   doctor = Yes (%)      51 (76.1)  47 (70.1)   0.135
#>   drink (%)                                    0.116
#>      Never drank         2 ( 3.0)   2 ( 3.0)
#>      Current drinker    54 (80.6)  51 (76.1)
#>      Former driker      11 (16.4)  14 (20.9)
#>   bp = Yes (%)           7 (10.4)   5 ( 7.5)   0.105
#>   immigrate (%)                                0.180
#>      not immigrant      64 (95.5)  61 (91.0)
#>      > 10 years          3 ( 4.5)   6 ( 9.0)
#>      recent              0 ( 0.0)   0 ( 0.0)
#>   fruit (%)                                    0.146
#>      0-3 daily serving  19 (28.4)  19 (28.4)
#>      4-6 daily serving  28 (41.8)  32 (47.8)
#>      6+ daily serving   20 (29.9)  16 (23.9)
#>   diab = Yes (%)         1 ( 1.5)   4 ( 6.0)   0.238
#>   edu (%)                                      0.105
```

```
#>      < 2ndary           14 (20.9)  13 (19.4)
#>      2nd grad.           1 ( 1.5)   2 ( 3.0)
#>      Other 2nd grad.     1 ( 1.5)   1 ( 1.5)
#>      Post-2nd grad.     51 (76.1)  51 (76.1)
```

Question for the students

- All SMD < 0.20?

Other balance measures

0.0.0.0.1 * Individual categories

If you want to check balance at each category (not very useful in general situations). We are generally interested if the variables are balanced or not (not categories).

```r
baltab <- bal.tab(matching.obj)
baltab
#> Call
#>  matchit(formula = ps.formula, data = analytic11n, method = "nearest",
#>     ratio = 1)
#>
#> Balance Measures
#>                              Type Diff.Adj
#> distance                 Distance   0.0597
#> age_20-29 years            Binary   0.0000
#> age_30-39 years            Binary  -0.0746
#> age_40-49 years            Binary   0.0299
#> age_50-59 years            Binary   0.0448
#> age_60-64 years            Binary   0.0000
#> sex_Male                   Binary   0.0597
#> stress_stressed            Binary   0.0149
#> married_single             Binary   0.0149
#> income_$29,999 or less     Binary   0.0299
#> income_$30,000-$49,999     Binary  -0.0597
#> income_$50,000-$79,999     Binary  -0.0299
#> income_$80,000 or more     Binary   0.0597
#> race_White                 Binary   0.0149
```

```
#> bmi_Underweight           Binary   0.0000
#> bmi_healthy weight        Binary   0.0000
#> bmi_Overweight            Binary   0.0000
#> phyact_Active             Binary   0.0000
#> phyact_Inactive           Binary  -0.0149
#> phyact_Moderate           Binary   0.0149
#> smoke_Never smoker        Binary  -0.0746
#> smoke_Current smoker      Binary   0.1045
#> smoke_Former smoker       Binary  -0.0299
#> doctor_Yes                Binary  -0.0597
#> drink_Never drank         Binary   0.0000
#> drink_Current drinker     Binary  -0.0448
#> drink_Former driker       Binary   0.0448
#> bp_Yes                    Binary  -0.0299
#> immigrate_not immigrant   Binary  -0.0448
#> immigrate_> 10 years      Binary   0.0448
#> immigrate_recent          Binary   0.0000
#> fruit_0-3 daily serving   Binary   0.0000
#> fruit_4-6 daily serving   Binary   0.0597
#> fruit_6+ daily serving    Binary  -0.0597
#> diab_Yes                  Binary   0.0448
#> edu_< 2ndary              Binary  -0.0149
#> edu_2nd grad.             Binary   0.0149
#> edu_Other 2nd grad.       Binary   0.0000
#> edu_Post-2nd grad.        Binary   0.0000
#>
#> Sample sizes
#>           Control Treated
#> All          1357      67
#> Matched        67      67
#> Unmatched    1290       0
```

0.0.0.0.2 * Individual plots

You could plot each variables individually

```r
bal.plot(matching.obj, var.name = "income")
```

```
bal.plot(matching.obj, var.name = "age")
```

```
bal.plot(matching.obj, var.name = "race")
```

```
bal.plot(matching.obj, var.name = "diab")
```

```
bal.plot(matching.obj, var.name = "immigrate")
```

0.0.0.0.3 * Love plot

```
# Individual categories again
love.plot(baltab, threshold = .2)
#> Warning: Unadjusted values are missing. This can occur when un = FALSE and
#> quick = TRUE in the original call to bal.tab().
#> Warning: Standardized mean differences and raw mean differences are present in the same plo
#> Use the 'stars' argument to distinguish between them and appropriately label the x-axis.
```

Repeat of Step 1-3 again

Covariate balance is reassessed in each step to ensure the quality of the match.

Add caliper

The matching process is repeated, this time introducing a caliper to ensure that matches are only made within a specified range of PS.

```r
logitPS <-  -log(1/OACVD.match$distance - 1)
# logit of the propensity score
.2*sd(logitPS) # suggested in the literature
#> [1] 0.2334615


# Step 1 and 2
matching.obj <- matchit(ps.formula,
                        data = analytic11n,
                        method = "nearest",
                        ratio = 1,
                        caliper = .2*sd(logitPS))
# see how many matched
matching.obj
#> A matchit object
#>  - method: 1:1 nearest neighbor matching without replacement
#>  - distance: Propensity score [caliper]
#>              - estimated with logistic regression
#>  - caliper: <distance> (0.015)
#>  - number of obs.: 1424 (original), 128 (matched)
#>  - target estimand: ATT
#>  - covariates: age, sex, stress, married, income, race, bmi, phyact, smoke, doctor, drink,
OACVD.match <- match.data(matching.obj)
```

```r
# Step 3
by(OACVD.match$distance, OACVD.match$OA, summary)
#> OACVD.match$OA: 0
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> 0.004671 0.041740 0.094963 0.124665 0.184103 0.418206
```

```
#> ------------------------------------------------------------
#> OACVD.match$OA: 1
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> 0.004671 0.041694 0.095089 0.125262 0.183739 0.424895
tab1 <- CreateTableOne(strata = "OA", data = OACVD.match,
                       test = FALSE, vars = var.names)
print(tab1, smd = TRUE)
#>                         Stratified by OA
#>                          0          1          SMD
#>   n                      64         64
#>   age (%)                                       0.196
#>      20-29 years          4 ( 6.2)   4 ( 6.2)
#>      30-39 years         16 (25.0)  11 (17.2)
#>      40-49 years         16 (25.0)  18 (28.1)
#>      50-59 years         25 (39.1)  28 (43.8)
#>      60-64 years          3 ( 4.7)   3 ( 4.7)
#>      65 years and over    0 ( 0.0)   0 ( 0.0)
#>      teen                 0 ( 0.0)   0 ( 0.0)
#>   sex = Male (%)         27 (42.2)  30 (46.9)   0.094
#>   stress = stressed (%) 18 (28.1)  18 (28.1)  <0.001
#>   married = single (%)  21 (32.8)  23 (35.9)   0.066
#>   income (%)                                    0.204
#>      $29,999 or less     11 (17.2)  13 (20.3)
#>      $30,000-$49,999     19 (29.7)  14 (21.9)
#>      $50,000-$79,999     13 (20.3)  12 (18.8)
#>      $80,000 or more     21 (32.8)  25 (39.1)
#>   race = White (%)       40 (62.5)  42 (65.6)   0.065
#>   bmi (%)                                      <0.001
#>      Underweight          0 ( 0.0)   0 ( 0.0)
#>      healthy weight      18 (28.1)  18 (28.1)
#>      Overweight          46 (71.9)  46 (71.9)
#>   phyact (%)                                    0.096
#>      Active              14 (21.9)  16 (25.0)
#>      Inactive            39 (60.9)  36 (56.2)
#>      Moderate            11 (17.2)  12 (18.8)
#>   smoke (%)                                     0.267
#>      Never smoker        14 (21.9)   9 (14.1)
#>      Current smoker      19 (29.7)  26 (40.6)
#>      Former smoker       31 (48.4)  29 (45.3)
#>   doctor = Yes (%)       48 (75.0)  44 (68.8)   0.139
```

```
#>   drink (%)                                    0.123
#>      Never drank         2 ( 3.1)   2 ( 3.1)
#>      Current drinker    52 (81.2)  49 (76.6)
#>      Former driker      10 (15.6)  13 (20.3)
#>   bp = Yes (%)           7 (10.9)   5 ( 7.8)   0.107
#>   immigrate (%)                                0.260
#>      not immigrant      62 (96.9)  58 (90.6)
#>      > 10 years          2 ( 3.1)   6 ( 9.4)
#>      recent              0 ( 0.0)   0 ( 0.0)
#>   fruit (%)                                    0.116
#>      0-3 daily serving  19 (29.7)  19 (29.7)
#>      4-6 daily serving  27 (42.2)  30 (46.9)
#>      6+ daily serving   18 (28.1)  15 (23.4)
#>   diab = Yes (%)         0 ( 0.0)   3 ( 4.7)   0.314
#>   edu (%)                                      0.108
#>      < 2ndary           14 (21.9)  13 (20.3)
#>      2nd grad.           1 ( 1.6)   2 ( 3.1)
#>      Other 2nd grad.     1 ( 1.6)   1 ( 1.6)
#>      Post-2nd grad.     48 (75.0)  48 (75.0)
```

Question for the students

- Did all of the SMDs decrease?

Look at the data

```r
# what is weights variable for pair matching?
head(OACVD.match)
#>            CVD         age    sex    married      race            edu
#> 17846 no event 30-39 years   Male     single     White Post-2nd grad.
#> 17864 no event 40-49 years Female not single Non-white       < 2ndary
#> 17904 no event 40-49 years Female not single Non-white       < 2ndary
#> 17909 no event 20-29 years Female not single     White Post-2nd grad.
#> 17921 no event 30-39 years Female     single     White Post-2nd grad.
#> 17953 no event 20-29 years   Male not single     White Post-2nd grad.
#>                income        bmi   phyact doctor           stress
#> 17846 $30,000-$49,999 Overweight Inactive     No Not too stressed
#> 17864 $30,000-$49,999 Overweight Inactive    Yes         stressed
```

```
#> 17904 $30,000-$49,999 Overweight Inactive    Yes Not too stressed
#> 17909 $80,000 or more Overweight Inactive    Yes Not too stressed
#> 17921 $29,999 or less Overweight Inactive     No         stressed
#> 17953 $50,000-$79,999 Overweight Moderate     No Not too stressed
#>                smoke           drink             fruit  bp diab province cycle
#> 17846   Never smoker Current drinker 0-3 daily serving  No   No    North    11
#> 17864 Current smoker Current drinker 4-6 daily serving  No   No    North    11
#> 17904 Current smoker   Former driker 0-3 daily serving Yes   No    North    11
#> 17909   Never smoker Current drinker 4-6 daily serving  No   No    North    11
#> 17921   Never smoker Current drinker 0-3 daily serving  No   No    North    11
#> 17953   Never smoker Current drinker 0-3 daily serving  No   No    North    11
#>          ID OA     immigrate miss survey.weight   distance weights subclass
#> 17846 17846  0 not immigrant    0         12.10 0.01854076       1       46
#> 17864 17864  1 not immigrant    0         38.23 0.18190027       1       30
#> 17904 17904  0 not immigrant    0         52.04 0.06783228       1       25
#> 17909 17909  0 not immigrant    0         33.23 0.03126371       1       36
#> 17921 17921  1 not immigrant    0         15.88 0.01857417       1       46
#> 17953 17953  0 not immigrant    0         26.91 0.00467081       1       24
summary(OACVD.match$weights)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>       1       1       1       1       1       1
```

Step 4

Estimate treatment effect for matched data

Different models (e.g., unconditional logistic regression, survey design) are fitted to estimate the treatment effect in the matched sample.

Unconditional logistic

```
# Wrong model for population!!
outcome.model <- glm(CVD ~ OA, data = OACVD.match, family = binomial())
publish(outcome.model)
#>  Variable Units OddsRatio       CI.95  p-value
#>        OA            0.48 [0.09;2.74]   0.4119
```

Survey design

0.0.0.0.1 * Convert data to design

The matched data is converted to a survey design object to account for the matched pairs in the analysis.

```
analytic.miss$matched <- 0
length(analytic.miss$ID) # full data
#> [1] 397173
length(OACVD.match$ID) # matched data
#> [1] 128
length(analytic.miss$ID[analytic.miss$ID %in% OACVD.match$ID])
#> [1] 128
analytic.miss$matched[analytic.miss$ID %in% OACVD.match$ID] <- 1
table(analytic.miss$matched)
#>
#>      0      1
#> 397045    128
w.design0 <- svydesign(id=~1, weights=~survey.weight,
                       data=analytic.miss)
w.design.m <- subset(w.design0, matched == 1)
```

0.0.0.0.2 * Balance in matched population?

```
tab1 <- svyCreateTableOne(strata = "OA", data = w.design.m,
                          test = FALSE, vars = var.names)
print(tab1, smd = TRUE)
#>                          Stratified by OA
#>                           0               1               SMD
#>   n                       1783.6          2002.1
#>   age (%)                                                  0.307
#>      20-29 years           131.0 ( 7.3)    120.9 ( 6.0)
#>      30-39 years           388.0 (21.8)    237.8 (11.9)
#>      40-49 years           518.3 (29.1)    572.7 (28.6)
#>      50-59 years           680.1 (38.1)    999.3 (49.9)
#>      60-64 years            66.1 ( 3.7)     71.4 ( 3.6)
#>      65 years and over       0.0 ( 0.0)      0.0 ( 0.0)
#>      teen                    0.0 ( 0.0)      0.0 ( 0.0)
#>   sex = Male (%)           852.4 (47.8)    985.1 (49.2)    0.028
```

```
#>   stress = stressed (%)  544.6 (30.5)   531.8 (26.6)   0.088
#>   married = single (%)   419.8 (23.5)   427.5 (21.4)   0.052
#>   income (%)                                           0.222
#>      $29,999 or less     266.6 (14.9)   352.4 (17.6)
#>      $30,000-$49,999     462.8 (25.9)   348.8 (17.4)
#>      $50,000-$79,999     298.6 (16.7)   315.7 (15.8)
#>      $80,000 or more     755.5 (42.4)   985.2 (49.2)
#>   race = White (%)      1129.2 (63.3)  1364.9 (68.2)   0.103
#>   bmi (%)                                              0.045
#>      Underweight           0.0 ( 0.0)     0.0 ( 0.0)
#>      healthy weight      483.3 (27.1)   583.3 (29.1)
#>      Overweight         1300.2 (72.9)  1418.8 (70.9)
#>   phyact (%)                                           0.054
#>      Active              448.0 (25.1)   493.9 (24.7)
#>      Inactive           1075.2 (60.3)  1176.6 (58.8)
#>      Moderate            260.3 (14.6)   331.5 (16.6)
#>   smoke (%)                                            0.288
#>      Never smoker        400.2 (22.4)   265.8 (13.3)
#>      Current smoker      548.8 (30.8)   836.6 (41.8)
#>      Former smoker       834.6 (46.8)   899.6 (44.9)
#>   doctor = Yes (%)      1376.0 (77.1)  1430.1 (71.4)   0.131
#>   drink (%)                                            0.194
#>      Never drank          44.0 ( 2.5)   112.6 ( 5.6)
#>      Current drinker    1464.1 (82.1)  1510.6 (75.5)
#>      Former driker       275.4 (15.4)   378.9 (18.9)
#>   bp = Yes (%)           166.6 ( 9.3)   153.5 ( 7.7)   0.060
#>   immigrate (%)                                        0.235
#>      not immigrant      1694.8 (95.0)  1774.1 (88.6)
#>      > 10 years           88.8 ( 5.0)   228.0 (11.4)
#>      recent                0.0 ( 0.0)     0.0 ( 0.0)
#>   fruit (%)                                            0.293
#>      0-3 daily serving   426.1 (23.9)   480.8 (24.0)
#>      4-6 daily serving   748.1 (41.9)  1082.3 (54.1)
#>      6+ daily serving    609.4 (34.2)   439.0 (21.9)
#>   diab = Yes (%)           0.0 ( 0.0)    83.6 ( 4.2)   0.295
#>   edu (%)                                              0.172
#>      < 2ndary            324.9 (18.2)   342.8 (17.1)
#>      2nd grad.            18.8 ( 1.1)    47.6 ( 2.4)
#>      Other 2nd grad.      15.2 ( 0.9)    52.2 ( 2.6)
#>      Post-2nd grad.     1424.7 (79.9)  1559.4 (77.9)
```

0.0.0.0.3 * Outcome analysis

```
fit.design <- svyglm(CVD ~ OA, design = w.design.m,
       family = binomial(logit))
#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
publish(fit.design)
#>  Variable Units OddsRatio       CI.95  p-value
#>        OA            0.50 [0.08;3.09]   0.4535
```

Matched data with increase ratio

The matching process is repeated with a different ratio (e.g., 1:5) to explore how changing the ratio affects the covariate balance and treatment effect estimation.

```
# Step 1 and 2
matching.obj <- matchit(ps.formula,
                        data = analytic11n,
                        method = "nearest",
                        ratio = 5,
                        caliper = 0.2)
# see how many matched
matching.obj
#> A matchit object
#>  - method: 5:1 nearest neighbor matching without replacement
#>  - distance: Propensity score [caliper]
#>              - estimated with logistic regression
#>  - caliper: <distance> (0.013)
#>  - number of obs.: 1424 (original), 349 (matched)
#>  - target estimand: ATT
#>  - covariates: age, sex, stress, married, income, race, bmi, phyact, smoke, doctor, drink,
OACVD.match <- match.data(matching.obj)
# Step 3
by(OACVD.match$distance, OACVD.match$OA, summary)
#> OACVD.match$OA: 0
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> 0.004643 0.039181 0.084421 0.101576 0.146403 0.418206
#> ------------------------------------------------------------
#> OACVD.match$OA: 1
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
```

```r
#> 0.004671 0.041694 0.095089 0.125262 0.183739 0.424895
tab1 <- CreateTableOne(strata = "OA", data = OACVD.match,
                       test = FALSE, vars = var.names)
print(tab1, smd = TRUE)
#>                          Stratified by OA
#>                           0           1           SMD
#>   n                       285         64
#>   age (%)                                          0.217
#>      20-29 years           24 ( 8.4)   4 ( 6.2)
#>      30-39 years           59 (20.7)  11 (17.2)
#>      40-49 years           94 (33.0)  18 (28.1)
#>      50-59 years           98 (34.4)  28 (43.8)
#>      60-64 years           10 ( 3.5)   3 ( 4.7)
#>      65 years and over      0 ( 0.0)   0 ( 0.0)
#>      teen                   0 ( 0.0)   0 ( 0.0)
#>   sex = Male (%)          132 (46.3)  30 (46.9)   0.011
#>   stress = stressed (%)    81 (28.4)  18 (28.1)   0.007
#>   married = single (%)     91 (31.9)  23 (35.9)   0.085
#>   income (%)                                       0.065
#>      $29,999 or less       64 (22.5)  13 (20.3)
#>      $30,000-$49,999       65 (22.8)  14 (21.9)
#>      $50,000-$79,999       51 (17.9)  12 (18.8)
#>      $80,000 or more      105 (36.8)  25 (39.1)
#>   race = White (%)        169 (59.3)  42 (65.6)   0.131
#>   bmi (%)                                          0.097
#>      Underweight            0 ( 0.0)   0 ( 0.0)
#>      healthy weight        68 (23.9)  18 (28.1)
#>      Overweight           217 (76.1)  46 (71.9)
#>   phyact (%)                                       0.136
#>      Active                57 (20.0)  16 (25.0)
#>      Inactive             178 (62.5)  36 (56.2)
#>      Moderate              50 (17.5)  12 (18.8)
#>   smoke (%)                                        0.097
#>      Never smoker          46 (16.1)   9 (14.1)
#>      Current smoker       123 (43.2)  26 (40.6)
#>      Former smoker        116 (40.7)  29 (45.3)
#>   doctor = Yes (%)        183 (64.2)  44 (68.8)   0.096
#>   drink (%)                                        0.051
#>      Never drank           10 ( 3.5)   2 ( 3.1)
#>      Current drinker      212 (74.4)  49 (76.6)
```

```
#>      Former driker       63 (22.1)  13 (20.3)
#>   bp = Yes (%)           22 ( 7.7)   5 ( 7.8)   0.003
#>   immigrate (%)                                 0.100
#>      not immigrant      266 (93.3)  58 (90.6)
#>      > 10 years          19 ( 6.7)   6 ( 9.4)
#>      recent               0 ( 0.0)   0 ( 0.0)
#>   fruit (%)                                     0.149
#>      0-3 daily serving  104 (36.5)  19 (29.7)
#>      4-6 daily serving  124 (43.5)  30 (46.9)
#>      6+ daily serving    57 (20.0)  15 (23.4)
#>   diab = Yes (%)         12 ( 4.2)   3 ( 4.7)   0.023
#>   edu (%)                                       0.137
#>      < 2ndary            67 (23.5)  13 (20.3)
#>      2nd grad.            9 ( 3.2)   2 ( 3.1)
#>      Other 2nd grad.      9 ( 3.2)   1 ( 1.6)
#>      Post-2nd grad.     200 (70.2)  48 (75.0)
```

0.0.0.0.1 * Question for the students

- Did all of the SMDs decrease?

0.0.0.0.2 * Look at the data

```
# what is weights variable now for 1:5 ratio?
head(OACVD.match)
#>            CVD         age    sex    married      race            edu
#> 17846 no event 30-39 years   Male     single     White Post-2nd grad.
#> 17847 no event 30-39 years   Male     single     White Post-2nd grad.
#> 17848    event 40-49 years   Male not single Non-white Post-2nd grad.
#> 17852 no event 40-49 years   Male     single     White Post-2nd grad.
#> 17863 no event 50-59 years Female not single Non-white       < 2ndary
#> 17864 no event 40-49 years Female not single Non-white       < 2ndary
#>                income        bmi   phyact doctor           stress
#> 17846 $30,000-$49,999 Overweight Inactive     No Not too stressed
#> 17847 $50,000-$79,999 Overweight Inactive    Yes Not too stressed
#> 17848 $30,000-$49,999 Overweight   Active     No         stressed
#> 17852 $50,000-$79,999 Overweight Moderate    Yes Not too stressed
#> 17863 $29,999 or less Overweight Inactive     No Not too stressed
#> 17864 $30,000-$49,999 Overweight Inactive    Yes         stressed
```

```
#>                 smoke           drink             fruit bp diab province cycle
#> 17846   Never smoker Current drinker 0-3 daily serving No   No    North    11
#> 17847 Current smoker Current drinker  6+ daily serving No   No    North    11
#> 17848 Current smoker Current drinker 0-3 daily serving No   No    North    11
#> 17852  Former smoker Current drinker 4-6 daily serving No   No    North    11
#> 17863 Current smoker     Never drank 0-3 daily serving No   No    North    11
#> 17864 Current smoker Current drinker 4-6 daily serving No   No    North    11
#>          ID OA     immigrate miss survey.weight   distance  weights subclass
#> 17846 17846  0 not immigrant    0         12.10 0.01854076 0.890625       46
#> 17847 17847  0 not immigrant    0         16.90 0.05730827 0.890625       41
#> 17848 17848  0 not immigrant    0         39.28 0.08589902 0.890625       63
#> 17852 17852  0 not immigrant    0         12.52 0.05208657 0.890625       12
#> 17863 17863  0 not immigrant    0         34.98 0.09000168 0.890625       43
#> 17864 17864  1 not immigrant    0         38.23 0.18190027 1.000000       30
summary(OACVD.match$weights)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.8906  0.8906  0.8906  1.0000  0.8906  4.4531
```

Combining matching weights

Different approaches to incorporating weights (e.g., matching weights, survey weights) are explored.

Not incorporating matching weights

```
analytic.miss$matched <- 0
length(analytic.miss$ID) # full data
#> [1] 397173
length(OACVD.match$ID) # matched data
#> [1] 349
length(analytic.miss$ID[analytic.miss$ID %in% OACVD.match$ID])
#> [1] 349
analytic.miss$matched[analytic.miss$ID %in% OACVD.match$ID] <- 1
table(analytic.miss$matched)
#>
#>      0      1
#> 396824    349
w.design0 <- svydesign(id=~1, weights=~survey.weight,
```

```
                        data=analytic.miss)
w.design.m <- subset(w.design0, matched == 1)
```

```
fit.design <- svyglm(CVD ~ OA, design = w.design.m,
       family = binomial(logit))
#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
publish(fit.design)
#>  Variable Units OddsRatio       CI.95 p-value
#>        OA            0.79 [0.23;2.80]   0.721
```

0.0.0.0.1 * Incorporating matching weights

```
analytic.miss$matched <- 0
length(analytic.miss$ID) # full data
#> [1] 397173
length(OACVD.match$ID) # matched data
#> [1] 349
length(analytic.miss$ID[analytic.miss$ID %in% OACVD.match$ID])
#> [1] 349
analytic.miss$matched[analytic.miss$ID %in% OACVD.match$ID] <- 1
table(analytic.miss$matched)
#>
#>      0      1
#> 396824    349
```

```
# multiply with matching (ratio) weights with survey weights
analytic.miss$combined.weight <- 0
analytic.miss$combined.weight[analytic.miss$ID %in% OACVD.match$ID] <-
  OACVD.match$weights*OACVD.match$survey.weight
w.design0 <- svydesign(id=~1, weights=~combined.weight,
                       data=analytic.miss)
w.design.m <- subset(w.design0, matched == 1)
```

```
fit.design <- svyglm(I(CVD=="event") ~ OA, design = w.design.m,
       family = binomial(logit))
#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
publish(fit.design)
#>  Variable Units OddsRatio       CI.95  p-value
#>        OA            1.14 [0.32;4.07]   0.8409
```

Matched with replacement

Matching is performed with replacement, allowing control units to be used in more than one match.

```
# Step 1 and 2
matching.obj <- matchit(ps.formula,
                        data = analytic11n,
                        method = "nearest",
                        ratio = 5,
                        caliper = 0.2,
                        replace = TRUE)
# see how many matched
matching.obj
#> A matchit object
#>  - method: 5:1 nearest neighbor matching with replacement
#>  - distance: Propensity score [caliper]
#>              - estimated with logistic regression
#>  - caliper: <distance> (0.013)
#>  - number of obs.: 1424 (original), 308 (matched)
#>  - target estimand: ATT
#>  - covariates: age, sex, stress, married, income, race, bmi, phyact, smoke, doctor, drink,
OACVD.match <- match.data(matching.obj)
# Step 3
by(OACVD.match$distance, OACVD.match$OA, summary)
#> OACVD.match$OA: 0
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> 0.004643 0.034244 0.067224 0.100845 0.148958 0.418206
#> ------------------------------------------------------------
#> OACVD.match$OA: 1
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> 0.004671 0.042322 0.097877 0.129743 0.189255 0.424895
tab1 <- CreateTableOne(strata = "OA", data = OACVD.match,
                       test = FALSE, vars = var.names)
print(tab1, smd = TRUE)
#>                        Stratified by OA
#>                         0           1          SMD
#>   n                     243         65
#>   age (%)                                       0.266
#>      20-29 years         22 ( 9.1)   4 ( 6.2)
#>      30-39 years         57 (23.5)  11 (16.9)
```

```
#>      40-49 years          74 (30.5)  18 (27.7)
#>      50-59 years          82 (33.7)  29 (44.6)
#>      60-64 years           8 ( 3.3)   3 ( 4.6)
#>      65 years and over     0 ( 0.0)   0 ( 0.0)
#>      teen                  0 ( 0.0)   0 ( 0.0)
#>   sex = Male (%)         113 (46.5)  31 (47.7)   0.024
#>   stress = stressed (%)  67 (27.6)  19 (29.2)   0.037
#>   married = single (%)   75 (30.9)  23 (35.4)   0.096
#>   income (%)                                    0.079
#>      $29,999 or less      53 (21.8)  13 (20.0)
#>      $30,000-$49,999      57 (23.5)  14 (21.5)
#>      $50,000-$79,999      44 (18.1)  12 (18.5)
#>      $80,000 or more      89 (36.6)  26 (40.0)
#>   race = White (%)       146 (60.1)  42 (64.6)   0.094
#>   bmi (%)                                       0.088
#>      Underweight           0 ( 0.0)   0 ( 0.0)
#>      healthy weight       58 (23.9)  18 (27.7)
#>      Overweight          185 (76.1)  47 (72.3)
#>   phyact (%)                                    0.160
#>      Active               45 (18.5)  16 (24.6)
#>      Inactive            155 (63.8)  37 (56.9)
#>      Moderate             43 (17.7)  12 (18.5)
#>   smoke (%)                                     0.095
#>      Never smoker         40 (16.5)   9 (13.8)
#>      Current smoker      101 (41.6)  26 (40.0)
#>      Former smoker       102 (42.0)  30 (46.2)
#>   doctor = Yes (%)       152 (62.6)  45 (69.2)   0.141
#>   drink (%)                                     0.059
#>      Never drank           9 ( 3.7)   2 ( 3.1)
#>      Current drinker     181 (74.5)  50 (76.9)
#>      Former driker        53 (21.8)  13 (20.0)
#>   bp = Yes (%)            19 ( 7.8)   5 ( 7.7)   0.005
#>   immigrate (%)                                 0.115
#>      not immigrant       228 (93.8)  59 (90.8)
#>      > 10 years           15 ( 6.2)   6 ( 9.2)
#>      recent                0 ( 0.0)   0 ( 0.0)
#>   fruit (%)                                     0.147
#>      0-3 daily serving    87 (35.8)  19 (29.2)
#>      4-6 daily serving   109 (44.9)  31 (47.7)
#>      6+ daily serving     47 (19.3)  15 (23.1)
```

```
#>   diab = Yes (%)          11 ( 4.5)   3 ( 4.6)   0.004
#>   edu (%)                                        0.175
#>      < 2ndary             58 (23.9)  13 (20.0)
#>      2nd grad.             8 ( 3.3)   2 ( 3.1)
#>      Other 2nd grad.       9 ( 3.7)   1 ( 1.5)
#>      Post-2nd grad.      168 (69.1)  49 (75.4)
```

Question for the students

- Did all of the SMDs decrease?

Look at the data

```
# what is weights variable now for 1:5 ratio?
head(OACVD.match)
#>            CVD         age    sex    married      race            edu
#> 17846 no event 30-39 years   Male     single     White Post-2nd grad.
#> 17847 no event 30-39 years   Male     single     White Post-2nd grad.
#> 17852 no event 40-49 years   Male     single     White Post-2nd grad.
#> 17863 no event 50-59 years Female not single Non-white       < 2ndary
#> 17864 no event 40-49 years Female not single Non-white       < 2ndary
#> 17904 no event 40-49 years Female not single Non-white       < 2ndary
#>                income        bmi   phyact doctor           stress
#> 17846 $30,000-$49,999 Overweight Inactive     No Not too stressed
#> 17847 $50,000-$79,999 Overweight Inactive    Yes Not too stressed
#> 17852 $50,000-$79,999 Overweight Moderate    Yes Not too stressed
#> 17863 $29,999 or less Overweight Inactive     No Not too stressed
#> 17864 $30,000-$49,999 Overweight Inactive    Yes         stressed
#> 17904 $30,000-$49,999 Overweight Inactive    Yes Not too stressed
#>                smoke           drink             fruit  bp diab province cycle
#> 17846   Never smoker Current drinker 0-3 daily serving  No   No    North    11
#> 17847 Current smoker Current drinker  6+ daily serving  No   No    North    11
#> 17852  Former smoker Current drinker 4-6 daily serving  No   No    North    11
#> 17863 Current smoker     Never drank 0-3 daily serving  No   No    North    11
#> 17864 Current smoker Current drinker 4-6 daily serving  No   No    North    11
#> 17904 Current smoker   Former driker 0-3 daily serving Yes   No    North    11
#>          ID OA     immigrate miss survey.weight   distance   weights
#> 17846 17846  0 not immigrant    0         12.10 0.01854076 0.7476923
```

```
#> 17847 17847  0 not immigrant    0         16.90 0.05730827 0.7476923
#> 17852 17852  0 not immigrant    0         12.52 0.05208657 1.4953846
#> 17863 17863  0 not immigrant    0         34.98 0.09000168 0.7476923
#> 17864 17864  1 not immigrant    0         38.23 0.18190027 1.0000000
#> 17904 17904  0 not immigrant    0         52.04 0.06783228 0.7476923
summary(OACVD.match$weights)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.7477  0.7477  0.7477  1.0000  1.0000  7.4769
```

Survey design

The matched data is converted into a survey design object, and the treatment effect is estimated while accounting for the complex survey design.

```
analytic.miss$matched <- 0
length(analytic.miss$ID) # full data
#> [1] 397173
length(OACVD.match$ID) # matched data
#> [1] 308
length(analytic.miss$ID[analytic.miss$ID %in% OACVD.match$ID])
#> [1] 308
analytic.miss$matched[analytic.miss$ID %in% OACVD.match$ID] <- 1
table(analytic.miss$matched)
#>
#>      0      1
#> 396865    308
```

```
# multiply with matching (ratio) weights with survey weights
analytic.miss$combined.weight <- 0
analytic.miss$combined.weight[analytic.miss$ID %in% OACVD.match$ID] <-
  OACVD.match$weights*OACVD.match$survey.weight
w.design0 <- svydesign(id=~1, weights=~combined.weight,
                       data=analytic.miss)
w.design.m <- subset(w.design0, matched == 1)
```

```
fit.design <- svyglm(I(CVD=="event") ~ OA, design = w.design.m,
       family = binomial(logit))
#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```r
publish(fit.design)
#>  Variable Units OddsRatio       CI.95  p-value
#>        OA            0.99 [0.26;3.73]   0.9932
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

PSM in OA-CVD (US)

Pre-processing

Load data

Load the dataset and inspect its structure and variables.

```
load(file="Data/propensityscore/NHANES17.RData")
ls()
#> [1] "analytic"           "analytic.with.miss" "has_annotations"
```

Visualize missing data patterns.

```
library(dplyr)
#>
#> Attaching package: 'dplyr'
#> The following objects are masked from 'package:data.table':
#>
#>     between, first, last
#> The following objects are masked from 'package:stats':
#>
#>     filter, lag
#> The following objects are masked from 'package:base':
#>
#>     intersect, setdiff, setequal, union
analytic.with.miss <- dplyr::select(analytic.with.miss,
                  cholesterol, #outcome
                  gender, age, born, race, education,
                  married, income, bmi, diabetes, #predictors
                  weight, psu, strata) #survey features

dim(analytic.with.miss)
#> [1] 9254   13
```

```r
str(analytic.with.miss)
#> 'data.frame':    9254 obs. of  13 variables:
#>  $ cholesterol: 'labelled' int  NA NA 157 148 189 209 176 NA 238 182 ...
#>   ..- attr(*, "label")= chr "Total Cholesterol (mg/dL)"
#>  $ gender     : chr  "Female" "Male" "Female" "Male" ...
#>  $ age        : 'labelled' int  NA NA 66 NA NA 66 75 NA 56 NA ...
#>   ..- attr(*, "label")= chr "Age in years at screening"
#>  $ born       : chr  "Born in 50 US states or Washingt" "Born in 50 US states or Washingt"
#>  $ race       : chr  "Other" "White" "Black" "Other" ...
#>  $ education  : chr  NA NA "High.School" NA ...
#>  $ married    : chr  NA NA "Previously.married" NA ...
#>  $ income     : chr  "Over100k" "Over100k" "<25k" NA ...
#>  $ bmi        : 'labelled' num  17.5 15.7 31.7 21.5 18.1 23.7 38.9 NA 21.3 19.7 ...
#>   ..- attr(*, "label")= chr "Body Mass Index (kg/m**2)"
#>  $ diabetes   : chr  "No" "No" "No" "No" ...
#>  $ weight     : 'labelled' num  8540 42567 8338 8723 7065 ...
#>   ..- attr(*, "label")= chr "Full sample 2 year MEC exam weight"
#>  $ psu        : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
#>  $ strata     : Factor w/ 15 levels "134","135","136",..: 12 10 12 1 5 5 3 1 1 14 ...
names(analytic.with.miss)
#>  [1] "cholesterol" "gender"      "age"         "born"        "race"
#>  [6] "education"   "married"     "income"      "bmi"         "diabetes"
#> [11] "weight"      "psu"         "strata"

library(DataExplorer)
plot_missing(analytic.with.miss)
```

Formatting variables

Rename variables to avoid conflicts. Recode variables into binary or categorical as needed. Ensure variable types (factor, numeric) are appropriate.

```
# to avaoid any confusion later
# rename weight variable as weights
# is reserved for matching weights
analytic.with.miss$survey.weight <- analytic.with.miss$weight
analytic.with.miss$weight <- NULL

#Creating binary variable for cholesterol
analytic.with.miss$cholesterol.bin <- ifelse(analytic.with.miss$cholesterol <200,
                                             1, #"healthy",
                                             0) #"unhealthy")
# exposure recoding
analytic.with.miss$diabetes <- ifelse(analytic.with.miss$diabetes == "Yes", 1, 0)

# ID
analytic.with.miss$ID <- 1:nrow(analytic.with.miss)

# covariates
analytic.with.miss$born <- ifelse(analytic.with.miss$born == "Other",
                                             0,
```

```
                                                 1)

vars = c("gender", "race", "education",
         "married", "income", "bmi")

numeric.names <- c("cholesterol", "bmi")
factor.names <- vars[!vars %in% numeric.names]

analytic.with.miss[factor.names] <- apply(X = analytic.with.miss[factor.names],
                                 MARGIN = 2, FUN = as.factor)

analytic.with.miss[numeric.names] <- apply(X = analytic.with.miss[numeric.names],
                                  MARGIN = 2, FUN =function (x)
                                    as.numeric(as.character(x)))
analytic.with.miss$income <- factor(analytic.with.miss$income,
                                      ordered = TRUE,
                                  levels = c("<25k", "Between.25kto54k",
                                             "Between.55kto99k",
                                             "Over100k"))

# features
table(analytic.with.miss$strata)
#>
#> 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148
#> 510 638 695 554 605 653 612 693 735 551 689 609 604 596 510
table(analytic.with.miss$psu)
#>
#>    1    2
#> 4464 4790
table(analytic.with.miss$strata,analytic.with.miss$psu)
#>
#>         1   2
#>   134 215 295
#>   135 316 322
#>   136 320 375
#>   137 306 248
#>   138 308 297
#>   139 278 375
#>   140 315 297
#>   141 282 411
```

```
#>   142 349 386
#>   143 232 319
#>   144 351 338
#>   145 339 270
#>   146 277 327
#>   147 335 261
#>   148 241 269

# impute
# require(mice)
# imputation1 <- mice(analytic.with.miss, seed = 123,
#                     m = 1, # Number of multiple imputations.
#                     maxit = 10 # Number of iteration; mostly useful for convergence
#                     )
# analytic.with.miss <- complete(imputation1)
# plot_missing(analytic.with.miss)
```

Complete case data

Create a dataset (`analytic.data`) without `NA` values for analysis. This is done for simplified analysis, but this approach has it's own challenges. In a next tutorial, we will appropriately deal with missing observations in a propensity score modelling.

```
dim(analytic.with.miss)
#> [1] 9254   15
analytic.data <- as.data.frame(na.omit(analytic.with.miss))
dim(analytic.data) # complete case
#> [1] 4167   15
```

Zanutto (2006)

- Ref: Zanutto (2006)

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Set seed

```
set.seed(123)
```

- "it is not necessary to use survey-weighted estimation for the propensity score model"

Propensity score analysis in 4 steps:

- Step 1: PS model specification
- Step 2: Matching based on the estimated propensity scores
- Step 3: Balance checking
- Step 4: Outcome modelling

Step 1

Specify the propensity score model to estimate propensity scores

```
ps.formula <- as.formula(diabetes ~ gender + born +
                         race + education + married + income + bmi)
```

Step 2

Match treated and untreated subjects based on the estimated propensity scores. Perform nearest-neighbor matching using the propensity scores. Visualize the distribution of propensity scores before and after matching.

```r
require(MatchIt)
set.seed(123)
# This function fits propensity score model (using logistic
# regression as above) when specified distance = 'logit'
# performs nearest-neighbor (NN) matching,
# without replacement
# with caliper = .2*SD of propensity score
# within which to draw control units
# with 1:1 ratio (pair-matching)
match.obj <- matchit(ps.formula, data = analytic.data,
                     distance = 'logit',
                     method = "nearest",
                     replace=FALSE,
                     caliper = .2,
                     ratio = 1)
# see matchit function options here
# https://www.rdocumentation.org/packages/MatchIt/versions/1.0-1/topics/matchit
analytic.data$PS <- match.obj$distance
summary(match.obj$distance)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> 0.02901 0.12255 0.17658 0.18982 0.23876 0.82164
plot(match.obj, type = "jitter")
```

```
#> [1] "To identify the units, use first mouse button; to stop, use second."
```

```
#> integer(0)
plot(match.obj, type = "hist")
```

```
tapply(analytic.data$PS, analytic.data$diabetes, summary)
#> $`0`
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> 0.02901 0.11509 0.16687 0.17949 0.22816 0.75968
#>
#> $`1`
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> 0.04793 0.16489 0.21300 0.23395 0.27768 0.82164
# check how many matched
match.obj
#> A matchit object
#>  - method: 1:1 nearest neighbor matching without replacement
#>  - distance: Propensity score [caliper]
#>              - estimated with logistic regression
#>  - caliper: <distance> (0.019)
#>  - number of obs.: 4167 (original), 1564 (matched)
#>  - target estimand: ATT
#>  - covariates: gender, born, race, education, married, income, bmi
# extract matched data
matched.data <- match.data(match.obj)
```

Step 3

Compare the similarity of baseline characteristics between treated and untreated subjects in a the propensity score-matched sample. In this case, we will compare SMD < 0.2 or not.

```r
require(tableone)
baselinevars <- c("gender", "born", "race", "education",
                  "married", "income", "bmi")
tab1 <- CreateTableOne(strata = "diabetes", vars = baselinevars,
                       data = analytic.data, test = FALSE)
print(tab1, smd = TRUE)
#>                        Stratified by diabetes
#>                         0             1             SMD
#>   n                      3376           791
#>   gender = Male (%)      1578 (46.7)    434 (54.9)   0.163
#>   born (mean (SD))       1.00 (0.00)   1.00 (0.00)  <0.001
#>   race (%)                                           0.060
#>      Black                728 (21.6)    183 (23.1)
#>      Hispanic             727 (21.5)    170 (21.5)
#>      Other                642 (19.0)    159 (20.1)
#>      White               1279 (37.9)    279 (35.3)
#>   education (%)                                      0.185
#>      College             1992 (59.0)    415 (52.5)
#>      High.School         1174 (34.8)    290 (36.7)
#>      School               210 ( 6.2)     86 (10.9)
#>   married (%)                                        0.316
#>      Married             2027 (60.0)    488 (61.7)
#>      Never.married        631 (18.7)     70 ( 8.8)
#>      Previously.married   718 (21.3)    233 (29.5)
#>   income (%)                                         0.092
#>      <25k                 830 (24.6)    225 (28.4)
#>      Between.25kto54k    1064 (31.5)    244 (30.8)
#>      Between.55kto99k     778 (23.0)    173 (21.9)
#>      Over100k             704 (20.9)    149 (18.8)
#>   bmi (mean (SD))       29.29 (7.11)  32.31 (8.03)   0.399
```

```r
tab1m <- CreateTableOne(strata = "diabetes", vars = baselinevars,
                        data = matched.data, test = FALSE)
```

```r
print(tab1m, smd = TRUE)
#>                          Stratified by diabetes
#>                           0              1              SMD
#>   n                         782            782
#>   gender = Male (%)         422 (54.0)     430 (55.0)    0.021
#>   born (mean (SD))         1.00 (0.00)    1.00 (0.00)   <0.001
#>   race (%)                                                0.068
#>      Black                  180 (23.0)     179 (22.9)
#>      Hispanic               151 (19.3)     170 (21.7)
#>      Other                  171 (21.9)     156 (19.9)
#>      White                  280 (35.8)     277 (35.4)
#>   education (%)                                           0.077
#>      College                441 (56.4)     411 (52.6)
#>      High.School            262 (33.5)     286 (36.6)
#>      School                  79 (10.1)      85 (10.9)
#>   married (%)                                             0.044
#>      Married                502 (64.2)     486 (62.1)
#>      Never.married           63 ( 8.1)      69 ( 8.8)
#>      Previously.married     217 (27.7)     227 (29.0)
#>   income (%)                                              0.065
#>      <25k                   202 (25.8)     218 (27.9)
#>      Between.25kto54k       236 (30.2)     244 (31.2)
#>      Between.55kto99k       187 (23.9)     171 (21.9)
#>      Over100k               157 (20.1)     149 (19.1)
#>   bmi (mean (SD))         32.12 (8.29)   32.05 (7.58)    0.009
```

Step 4

Estimate the effect of treatment on outcomes using propensity score-matched sample. Use the matched sample to estimate the treatment effect, considering survey design.

Incorporating the survey design into both linear regression and propensity score analysis is crucial. Neglecting the survey weights can significantly impact the estimates, altering the representation of population-level effects.

```r
require(survey)
# setup the design with survey features
analytic.with.miss$matched <- 0
```

```r
length(analytic.with.miss$ID) # full data
#> [1] 9254
length(matched.data$ID) # matched data
#> [1] 1564
length(analytic.with.miss$ID[analytic.with.miss$ID %in% matched.data$ID])
#> [1] 1564
analytic.with.miss$matched[analytic.with.miss$ID %in% matched.data$ID] <- 1
table(analytic.with.miss$matched)
#>
#>    0    1
#> 7690 1564
w.design0 <- svydesign(strata=~strata, id=~psu, weights=~survey.weight,
                       data=analytic.with.miss, nest=TRUE)
w.design.m <- subset(w.design0, matched == 1)
```

```r
out.formula <- as.formula(cholesterol.bin ~ diabetes)
sfit <- svyglm(out.formula,family=binomial(logit), design = w.design.m)
#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
require(jtools)
summ(sfit, exp = TRUE, confint = TRUE)
```

Observations	1564
Dependent variable	cholesterol.bin
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R² (Cragg-Uhler)	0.02
Pseudo-R² (McFadden)	0.01
AIC	1924.27

	exp(Est.)	2.5%	97.5%	t val.	p
(Intercept)	1.34	0.98	1.83	1.84	0.09
diabetes	1.67	1.17	2.37	2.84	0.01

Standard errors: Robust

DuGoff et al. (2014)

- Ref: DuGoff, Schuler, and Stuart (2014)

Propensity score analysis in 4 steps (PATT)

Video content (optional)

> Tip
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Step 1

Specify the propensity score model to estimate propensity scores. Similar to Zanutto but includes additional covariates in the model.

```
# response = exposure variable
# independent variables = baseline covariates
ps.formula <- as.formula(diabetes ~ gender + born + race + education +
                            married + income + bmi+
                           psu+strata+survey.weight)
```

Step 2

Match treated and untreated subjects on the estimated propensity scores

```
require(MatchIt)
set.seed(123)
match.obj <- matchit(ps.formula, data = analytic.data,
                     distance = 'logit',
                     method = "nearest",
                     replace=FALSE,
                     caliper = .2,
```

```r
                        ratio = 1)
analytic.data$PS <- match.obj$distance
summary(match.obj$distance)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> 0.00363 0.11341 0.17509 0.18982 0.24765 0.80853
plot(match.obj, type = "jitter")
```

```
#> [1] "To identify the units, use first mouse button; to stop, use second."
#> integer(0)
plot(match.obj, type = "hist")
```

```r
tapply(analytic.data$PS, analytic.data$diabetes, summary)
#> $`0`
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> 0.00363 0.10394 0.16243 0.17690 0.23461 0.73143
#>
#> $`1`
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> 0.01182 0.17245 0.22748 0.24500 0.29948 0.80853
# check how many matched
match.obj
#> A matchit object
#>  - method: 1:1 nearest neighbor matching without replacement
#>  - distance: Propensity score [caliper]
#>              - estimated with logistic regression
#>  - caliper: <distance> (0.021)
#>  - number of obs.: 4167 (original), 1570 (matched)
#>  - target estimand: ATT
#>  - covariates: gender, born, race, education, married, income, bmi, psu, strata, survey.wei
# extract matched data
matched.data <- match.data(match.obj)
```

Step 3

Compare the similarity of baseline characteristics between treated and untreated subjects in a the propensity score-matched sample. In this case, we will compare SMD < 0.2 or not.

```r
require(tableone)
baselinevars <- c("gender", "born", "race", "education",
                  "married", "income", "bmi",
                  "psu", "strata", "survey.weight")
matched.data$survey.weight <- as.numeric(as.character(matched.data$survey.weight))
matched.data$strata <- as.numeric(as.character(matched.data$strata))
tab1m <- CreateTableOne(strata = "diabetes", vars = baselinevars,
                        data = matched.data, test = FALSE)
print(tab1m, smd = TRUE)
#>                            Stratified by diabetes
#>                             0                   1                   SMD
#>   n                              785                 785
#>   gender = Male (%)              433 (55.2)          431 (54.9)       0.005
#>   born (mean (SD))              1.00 (0.00)         1.00 (0.00)      <0.001
#>   race (%)                                                            0.048
#>      Black                       193 (24.6)          180 (22.9)
#>      Hispanic                    163 (20.8)          170 (21.7)
#>      Other                       163 (20.8)          158 (20.1)
#>      White                       266 (33.9)          277 (35.3)
#>   education (%)                                                       0.032
#>      College                     403 (51.3)          412 (52.5)
#>      High.School                 300 (38.2)          288 (36.7)
#>      School                       82 (10.4)           85 (10.8)
#>   married (%)                                                         0.030
#>      Married                     473 (60.3)          484 (61.7)
#>      Never.married                71 ( 9.0)           70 ( 8.9)
#>      Previously.married          241 (30.7)          231 (29.4)
#>   income (%)                                                          0.035
#>      <25k                        232 (29.6)          222 (28.3)
#>      Between.25kto54k            236 (30.1)          242 (30.8)
#>      Between.55kto99k            176 (22.4)          173 (22.0)
#>      Over100k                    141 (18.0)          148 (18.9)
#>   bmi (mean (SD))               31.92 (8.33)        32.09 (7.52)      0.020
#>   psu = 2 (%)                    382 (48.7)          394 (50.2)       0.031
```

```
#>   strata (mean (SD))          140.84 (4.24)       140.97 (4.23)      0.031
#>   survey.weight (mean (SD)) 35647.01 (37699.98) 35596.81 (45212.82)  0.001
```

Step 4

Estimate the effect of treatment on outcomes using propensity score-matched sample

```
# setup the design with survey features
analytic.with.miss$matched <- 0
length(analytic.with.miss$ID) # full data
#> [1] 9254
length(matched.data$ID) # matched data
#> [1] 1570
length(analytic.with.miss$ID[analytic.with.miss$ID %in% matched.data$ID])
#> [1] 1570
analytic.with.miss$matched[analytic.with.miss$ID %in% matched.data$ID] <- 1
table(analytic.with.miss$matched)
#>
#>    0    1
#> 7684 1570
w.design0 <- svydesign(strata=~strata, id=~psu, weights=~survey.weight,
                       data=analytic.with.miss, nest=TRUE)
w.design.m <- subset(w.design0, matched == 1)
```

```
out.formula <- as.formula(cholesterol.bin ~ diabetes)
sfit <- svyglm(out.formula,family=binomial, design = w.design.m)
#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
require(jtools)
summ(sfit, exp = TRUE, confint = TRUE)
```

Observations	1570
Dependent variable	cholesterol.bin
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R² (Cragg-Uhler)	0.01
Pseudo-R² (McFadden)	0.00
AIC	1918.09

	exp(Est.)	2.5%	97.5%	t val.	p
(Intercept)	1.69	1.33	2.15	4.24	0.00
diabetes	1.32	0.99	1.77	1.86	0.08

Standard errors: Robust

Austin et al. (2018)

- Ref: Peter C. Austin, Jembere, and Chiu (2018)

Propensity score analysis in 4 steps (PATT)

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Step 1

Specify the propensity score model to estimate propensity scores. Use survey logistic regression to account for survey design in propensity score estimation.

```r
# response = exposure variable
# independent variables = baseline covariates
ps.formula <- as.formula(diabetes ~ gender + born + race + education +
                            married + income + bmi)
require(survey)
analytic.design <- svydesign(id=~psu,weights=~survey.weight,
                             strata=~strata,
                             data=analytic.data, nest=TRUE)
```

```r
ps.fit <- svyglm(ps.formula, design=analytic.design, family=quasibinomial)
analytic.data$PS <- fitted(ps.fit)
summary(analytic.data$PS)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> 0.01352 0.08788 0.13399 0.15120 0.19285 0.86375
```

Step 2

Match treated and untreated subjects on the estimated propensity scores. Two methods are explored: using the `Matching` package and the `MatchIt` package.

```r
require(Matching)
#> Loading required package: Matching
#> Warning: package 'Matching' was built under R version 4.2.3
#> Loading required package: MASS
#>
#> Attaching package: 'MASS'
#> The following object is masked from 'package:dplyr':
#>
#>     select
#> ##
#> ##  Matching (Version 4.10-14, Build Date: 2023-09-13)
#> ##  See https://www.jsekhon.com for additional documentation.
#> ##  Please cite software as:
#> ##   Jasjeet S. Sekhon. 2011. ``Multivariate and Propensity Score Matching
#> ##   Software with Automated Balance Optimization: The Matching package for R.''
#> ##   Journal of Statistical Software, 42(7): 1-52.
#> ##
match.obj2 <- Match(Y=analytic.data$cholesterol,
                    Tr=analytic.data$diabetes,
                    X=analytic.data$PS,
                    M=1,
                    estimand = "ATT",
                    replace=FALSE,
                    caliper = 0.2)
summary(match.obj2)
#>
#> Estimate...  -15.287
```

```r
#> SE.........  2.118
#> T-stat.....  -7.2175
#> p.val......  5.2958e-13
#>
#> Original number of observations..............  4167
#> Original number of treated obs...............  791
#> Matched number of observations...............  781
#> Matched number of observations  (unweighted).  781
#>
#> Caliper (SDs)........................................   0.2
#> Number of obs dropped by 'exact' or 'caliper'  10
matched.data2 <- analytic.data[c(match.obj2$index.treated,
                                 match.obj2$index.control),]
dim(matched.data2)
#> [1] 1562   16
```

```r
require(MatchIt)
set.seed(123)
match.obj <- matchit(ps.formula, data = analytic.data,
                     distance = analytic.data$PS,
                     method = "nearest",
                     replace=FALSE,
                     caliper = .2,
                     ratio = 1)
analytic.data$PS <- match.obj$distance
summary(match.obj$distance)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> 0.01352 0.08788 0.13399 0.15120 0.19285 0.86375
plot(match.obj, type = "jitter")
```

```
#> [1] "To identify the units, use first mouse button; to stop, use second."
#> integer(0)
plot(match.obj, type = "hist")
```

```
tapply(analytic.data$PS, analytic.data$diabetes, summary)
#> $`0`
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> 0.01352 0.08182 0.12644 0.14119 0.18267 0.75047
#>
```

```
#> $`1`
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> 0.02352 0.12362 0.16998 0.19389 0.23171 0.86375
# check how many matched
match.obj
#> A matchit object
#>  - method: 1:1 nearest neighbor matching without replacement
#>  - distance: User-defined [caliper]
#>  - caliper: <distance> (0.019)
#>  - number of obs.: 4167 (original), 1568 (matched)
#>  - target estimand: ATT
#>  - covariates: gender, born, race, education, married, income, bmi
# extract matched data
matched.data2 <- match.data(match.obj)
dim(matched.data2)
#> [1] 1568   19
```

Step 3

Compare the similarity of baseline characteristics between treated and untreated subjects in a the propensity score-matched sample. In this case, we will compare SMD < 0.2 or not.

```
baselinevars <- c("gender", "born", "race", "education",
                  "married", "income", "bmi")
tab1m <- CreateTableOne(strata = "diabetes",
                        vars = baselinevars,
                        data = matched.data2, test = FALSE)
print(tab1m, smd = TRUE)
#>                        Stratified by diabetes
#>                         0             1             SMD
#>   n                       784           784
#>   gender = Male (%)       405 (51.7)    431 (55.0)   0.067
#>   born (mean (SD))       1.00 (0.00)   1.00 (0.00)  <0.001
#>   race (%)                                           0.134
#>      Black                163 (20.8)    182 (23.2)
#>      Hispanic             139 (17.7)    170 (21.7)
#>      Other                171 (21.8)    157 (20.0)
```

```
#>      White                 311 (39.7)    275 (35.1)
#>   education (%)                                     0.040
#>      College               428 (54.6)    413 (52.7)
#>      High.School           274 (34.9)    288 (36.7)
#>      School                 82 (10.5)     83 (10.6)
#>   married (%)                                       0.070
#>      Married               509 (64.9)    485 (61.9)
#>      Never.married          59 ( 7.5)     70 ( 8.9)
#>      Previously.married    216 (27.6)    229 (29.2)
#>   income (%)                                        0.063
#>      <25k                  220 (28.1)    220 (28.1)
#>      Between.25kto54k      226 (28.8)    242 (30.9)
#>      Between.55kto99k      192 (24.5)    173 (22.1)
#>      Over100k              146 (18.6)    149 (19.0)
#>   bmi (mean (SD))        31.93 (8.16)  32.11 (7.61)   0.023
```

Step 4

Estimate the effect of treatment on outcomes using propensity score-matched sample.

```r
# setup the design with survey features
analytic.with.miss$matched <- 0
length(analytic.with.miss$ID) # full data
#> [1] 9254
length(matched.data2$ID) # matched data
#> [1] 1568
length(analytic.with.miss$ID[analytic.with.miss$ID %in% matched.data2$ID])
#> [1] 1568
analytic.with.miss$matched[analytic.with.miss$ID %in% matched.data2$ID] <- 1
table(analytic.with.miss$matched)
#>
#>    0    1
#> 7686 1568
w.design0 <- svydesign(strata=~strata, id=~psu, weights=~survey.weight,
                       data=analytic.with.miss, nest=TRUE)
w.design.m2 <- subset(w.design0, matched == 1)
```

```r
out.formula <- as.formula(cholesterol.bin ~ diabetes)
sfit <- svyglm(out.formula,family=binomial, design = w.design.m2)
#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!

require(jtools)
summ(sfit, exp = TRUE, confint = TRUE)
```

Observations	1568
Dependent variable	cholesterol.bin
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R^2 (Cragg-Uhler)	0.01
Pseudo-R^2 (McFadden)	0.01
AIC	1918.99

	exp(Est.)	2.5%	97.5%	t val.	p
(Intercept)	1.47	1.13	1.91	2.88	0.01
diabetes	1.51	1.20	1.89	3.58	0.00

Standard errors: Robust

PSM in BMI-diabetes

Propensity analysis problem

- Perform a **propensity score matching** (pair matching, without replacement, using SMD cut-point 0.2, may adjust for imbalanced and/or all covariates in the outcome model, if any) analysis as per the following recommendations
  - Zanutto (2006)
  - DuGoff, Schuler, and Stuart (2014)
  - Peter C. Austin, Jembere, and Chiu (2018)

See explained in the previous chapter on PSM in OA-CVD (US), there are four steps of the propensity score matching:

- Step 1: PS model specification
- Step 2: Matching based on the estimated propensity scores
- Step 3: Balance checking
- Step 4: Outcome modelling

Only Step 1 is different for Zanutto (2006), DuGoff et al. (2014), and Austin et al. (2018) approaches.

Dataset

- The following modified NHANES dataset
  - NHANES15lab5.RData
- use the data set "analytic.with.miss" within this file.
  - for obtaining the final treatment effect estimates, you can omit missing values, but only after creating the design (e.g., subset the design, not the data itself directly).

Variables

- Outcome: diabetes
  - 'No' as the reference category
- Exposure: bmi
  - convert to binary with >25 vs. <= 25,
  - with > 25 as the reference category
- Confounder list:
  - gender
  - age
    - assume continuous
  - race
  - income
  - education
  - married
  - cholesterol
  - diastolicBP
  - systolicBP
- Mediator:
  - physical.work
    - 'No' as the reference category
- Survey features
  - psu
  - strata
  - weight

Pre-processing

Load data

```
load(file="Data/propensityscore/NHANES15lab5.RData")
```

Variable summary

```r
# Exposure
analytic.with.miss$bmi <- with(analytic.with.miss,
                                  ifelse(bmi>25, "Overweight",
                                         ifelse(bmi<=25, "Not overweight", NA)))
analytic.with.miss$bmi <- as.factor(analytic.with.miss$bmi)
analytic.with.miss$bmi <- relevel(analytic.with.miss$bmi, ref = "Overweight")

# Drop unnecessary variables
analytic.with.miss$born <- NULL
analytic.with.miss$physical.work <- NULL

# Rename the weight variable into interview.weight
names(analytic.with.miss)[names(analytic.with.miss) == "weight"] <- "interview.weight"
```

Reproducibility

```r
set.seed(504)
```

Approach by Zanutto (2006)

Step 1

```r
# Specify the PS model to estimate propensity scores
ps.formula <- as.formula(I(bmi=="Not overweight") ~ gender + age + race + income + education +
                           married + cholesterol + diastolicBP + systolicBP)
```

Step 2

```r
require(MatchIt)
# Complete data for matching
analytic.data <- analytic.with.miss[complete.cases(analytic.with.miss),]

```

```r
# Create a missing variable in the original dataset
analytic.with.miss$miss <- 1
analytic.with.miss$miss[analytic.with.miss$ID %in% analytic.data$ID] <- 0

# Propensity scores
ps.fit <- glm(ps.formula, data = analytic.data, family = binomial("logit"))
analytic.data$PS <- predict(ps.fit, type = "response", newdata = analytic.data)

# Caliper fixing to 0.2*sd(logit of PS)
caliper <- 0.2*sd(log(analytic.data$PS/(1-analytic.data$PS)))

# Match exposed and unexposed subjects
match.obj <- matchit(ps.formula, data = analytic.data,
                     distance = analytic.data$PS,
                     method = "nearest",
                     replace = FALSE,
                     caliper = caliper,
                     ratio = 1)
analytic.data$PS <- match.obj$distance

# Extract matched data
matched.data <- match.data(match.obj)
```

Step 3

```r
# Balance checking
cov <- c("gender", "age", "race", "income",
         "education", "married", "cholesterol",
         "diastolicBP", "systolicBP")

tab1m <- CreateTableOne(strata = "bmi", vars = cov, data = matched.data, test = F)
print(tab1m, smd = TRUE)
#>                          Stratified by bmi
#>                           Overweight     Not overweight SMD
#>   n                         2047           2047
#>   gender = Male (%)          987 (48.2)     993 (48.5)   0.006
#>   age (mean (SD))          46.96 (17.55)  46.35 (17.64)  0.035
#>   race (%)                                               0.041
```

```
#>      Black                   452 (22.1)     424 (20.7)
#>      Hispanic                558 (27.3)     584 (28.5)
#>      Other                   328 (16.0)     338 (16.5)
#>      White                   709 (34.6)     701 (34.2)
#>   income (%)                                            0.042
#>      <25k                    537 (26.2)     509 (24.9)
#>      Between.25kto54k        670 (32.7)     661 (32.3)
#>      Between.55kto99k        472 (23.1)     484 (23.6)
#>      Over100k                368 (18.0)     393 (19.2)
#>   education (%)                                         0.023
#>      College                1199 (58.6)    1202 (58.7)
#>      High.School             667 (32.6)     652 (31.9)
#>      School                  181 ( 8.8)     193 ( 9.4)
#>   married (%)                                           0.036
#>      Married                1215 (59.4)    1243 (60.7)
#>      Never.married           402 (19.6)     374 (18.3)
#>      Previously.married      430 (21.0)     430 (21.0)
#>   cholesterol (mean (SD)) 176.45 (39.57) 174.62 (38.70)  0.047
#>   diastolicBP (mean (SD))  64.97 (13.65)  64.19 (13.47)  0.057
#>   systolicBP (mean (SD))  118.07 (15.23) 116.19 (17.60)  0.114
```

All SMDs are less than your specified cut-point? If yes, that indicates that there is good covariate balancing.

Step 4

```r
require(survey)
# Setup the design with survey features
analytic.with.miss$matched <- 0
analytic.with.miss$matched[analytic.with.miss$ID %in% matched.data$ID] <- 1

# Survey setup for full data
w.design0 <- svydesign(strata = ~strata, id = ~psu, weights = ~interview.weight,
                       data = analytic.with.miss, nest = TRUE)

# Subset matched data
w.design.m <- subset(w.design0, matched == 1)

```

```r
# Outcome model
out.formula <- as.formula(I(diabetes == "Yes") ~ bmi)
fit <- svyglm(out.formula, design = w.design.m, family = binomial("logit"))
require(jtools)
summ(fit, exp = TRUE, confint = TRUE, digits = 3)
```

Observations	4094
Dependent variable	I(diabetes == "Yes")
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R^2 (Cragg-Uhler)	0.071
Pseudo-R^2 (McFadden)	0.054
AIC	2336.725

	exp(Est.)	2.5%	97.5%	t val.	p
(Intercept)	0.165	0.141	0.195	-21.678	0.000
bmiNot overweight	0.245	0.173	0.346	-7.980	0.000

Standard errors: Robust

Double adjustment

```r
library(survey)
# Outcome model with covariates adjustment
fit.DA <- svyglm(I(diabetes == "Yes") ~ bmi + gender + age + race + income +
                 education + married + cholesterol + diastolicBP + systolicBP,
               design = w.design.m, family = binomial("logit"))
summ(fit.DA, exp = TRUE, confint = TRUE, digits = 3)
```

Observations	4094
Dependent variable	I(diabetes == ”Yes”)
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R² (Cragg-Uhler)	0.212
Pseudo-R² (McFadden)	0.166
AIC	2063.466

	exp(Est.)	2.5%	97.5%	t val.	p
(Intercept)	0.045	0.010	0.199	-4.101	NA
bmiNot overweight	0.235	0.159	0.348	-7.219	NA
genderMale	1.317	1.082	1.604	2.740	NA
age	1.051	1.044	1.059	13.280	NA
raceHispanic	0.896	0.649	1.237	-0.665	NA
raceOther	1.158	0.565	2.374	0.401	NA
raceWhite	0.778	0.507	1.193	-1.152	NA
incomeBetween.25kto54k	0.830	0.551	1.251	-0.889	NA
incomeBetween.55kto99k	0.872	0.621	1.224	-0.794	NA
incomeOver100k	0.669	0.418	1.072	-1.669	NA
educationHigh.School	0.948	0.733	1.225	-0.408	NA
educationSchool	0.872	0.448	1.695	-0.405	NA
marriedNever.married	1.055	0.677	1.646	0.237	NA
marriedPreviously.married	0.849	0.534	1.348	-0.694	NA
cholesterol	0.991	0.986	0.995	-3.810	NA
diastolicBP	1.005	0.992	1.018	0.749	NA
systolicBP	1.003	0.993	1.012	0.559	NA

Standard errors: Robust

Approach by DuGoff et al. (2014)

Step 1

```
# Specify the PS model to estimate propensity scores
ps.formula2 <- as.formula(I(bmi == "Not overweight") ~ gender + age + race + income + educatio
```

```
                              married + cholesterol + diastolicBP + systolicBP +
                              psu + strata + interview.weight)
```

Step 2

```
# Propensity scores
ps.fit2 <- glm(ps.formula2, data = analytic.data, family = binomial("logit"))
analytic.data$PS2 <- fitted(ps.fit2)

# Caliper fixing to 0.2*sd(logit of PS)
caliper2 <- 0.2*sd(log(analytic.data$PS2/(1-analytic.data$PS2)))

# Match exposed and unexposed subjects
set.seed(504)
match.obj2 <- matchit(ps.formula2, data = analytic.data,
                      distance = analytic.data$PS2,
                      method = "nearest",
                      replace = FALSE,
                      caliper = caliper2,
                      ratio = 1)
analytic.data$PS2 <- match.obj2$distance

# Extract matched data
matched.data2 <- match.data(match.obj2)
```

Step 3

```
# Balance checking
cov2 <- c("gender", "age", "race", "income",
          "education", "married", "cholesterol",
         "diastolicBP", "systolicBP",
         "psu", "strata", "interview.weight")

tab1m2 <- CreateTableOne(strata = "bmi", vars = cov2, data = matched.data2, test = F)
print(tab1m2, smd = TRUE)
#>                                Stratified by bmi
```

```
#>                                  Overweight          Not overweight      SMD
#>   n                                  2037                2037
#>   gender = Male (%)                   991 (48.6)          985 (48.4)      0.006
#>   age (mean (SD))                   47.10 (17.50)       46.41 (17.72)     0.039
#>   race (%)                                                                  0.044
#>      Black                            453 (22.2)          438 (21.5)
#>      Hispanic                         553 (27.1)          579 (28.4)
#>      Other                            324 (15.9)          342 (16.8)
#>      White                            707 (34.7)          678 (33.3)
#>   income (%)                                                                0.023
#>      <25k                             522 (25.6)          522 (25.6)
#>      Between.25kto54k                 679 (33.3)          662 (32.5)
#>      Between.55kto99k                 471 (23.1)          473 (23.2)
#>      Over100k                         365 (17.9)          380 (18.7)
#>   education (%)                                                             0.007
#>      College                         1199 (58.9)         1192 (58.5)
#>      High.School                      644 (31.6)          648 (31.8)
#>      School                           194 ( 9.5)          197 ( 9.7)
#>   married (%)                                                               0.018
#>      Married                         1209 (59.4)         1224 (60.1)
#>      Never.married                    395 (19.4)          394 (19.3)
#>      Previously.married               433 (21.3)          419 (20.6)
#>   cholesterol (mean (SD))         176.73 (39.68)      174.82 (38.77)     0.049
#>   diastolicBP (mean (SD))          65.16 (13.49)       64.39 (13.22)     0.057
#>   systolicBP (mean (SD))          118.27 (15.07)      116.33 (17.57)     0.119
#>   psu (mean (SD))                   1.49 (0.50)         1.49 (0.50)      0.008
#>   strata (mean (SD))              126.39 (4.16)       126.27 (4.29)      0.029
#>   interview.weight (mean (SD)) 38621.02 (34640.92) 37195.99 (36180.33)  0.040
```

All SMDs are less than your specified cut-point? If yes, that indicates that there is good covariate balancing.

Step 4

```
# Setup the design with survey features
analytic.with.miss$matched2 <- 0
analytic.with.miss$matched2[analytic.with.miss$ID %in% matched.data2$ID] <- 1

```

```r
# Survey setup for full data
w.design02 <- svydesign(strata = ~strata, id = ~psu, weights = ~interview.weight,
                        data = analytic.with.miss, nest = TRUE)

# Subset matched data
w.design.m2 <- subset(w.design02, matched2 == 1)

# Outcome model
out.formula <- as.formula(I(diabetes == "Yes") ~ bmi)
fit2 <- svyglm(out.formula, design = w.design.m2, family = binomial("logit"))
summ(fit2, exp = TRUE, confint = TRUE, digits = 3)
```

Observations	4074
Dependent variable	I(diabetes == "Yes")
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R^2 (Cragg-Uhler)	0.083
Pseudo-R^2 (McFadden)	0.063
AIC	2322.836

	exp(Est.)	2.5%	97.5%	t val.	p
(Intercept)	0.174	0.149	0.202	-22.325	0.000
bmiNot overweight	0.221	0.163	0.299	-9.749	0.000

Standard errors: Robust

Double adjustment

```r
# Outcome model with covariates adjustment
fit2.DA <- svyglm(I(diabetes == "Yes") ~ bmi + gender + age + race + income +
                  education + married + cholesterol + diastolicBP + systolicBP,
                design = w.design.m, family = binomial("logit"))
summ(fit2.DA, exp = TRUE, confint = TRUE, digits = 3)
```

Observations	4094
Dependent variable	I(diabetes == ”Yes”)
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R² (Cragg-Uhler)	0.212
Pseudo-R² (McFadden)	0.166
AIC	2063.466

	exp(Est.)	2.5%	97.5%	t val.	p
(Intercept)	0.045	0.010	0.199	-4.101	NA
bmiNot overweight	0.235	0.159	0.348	-7.219	NA
genderMale	1.317	1.082	1.604	2.740	NA
age	1.051	1.044	1.059	13.280	NA
raceHispanic	0.896	0.649	1.237	-0.665	NA
raceOther	1.158	0.565	2.374	0.401	NA
raceWhite	0.778	0.507	1.193	-1.152	NA
incomeBetween.25kto54k	0.830	0.551	1.251	-0.889	NA
incomeBetween.55kto99k	0.872	0.621	1.224	-0.794	NA
incomeOver100k	0.669	0.418	1.072	-1.669	NA
educationHigh.School	0.948	0.733	1.225	-0.408	NA
educationSchool	0.872	0.448	1.695	-0.405	NA
marriedNever.married	1.055	0.677	1.646	0.237	NA
marriedPreviously.married	0.849	0.534	1.348	-0.694	NA
cholesterol	0.991	0.986	0.995	-3.810	NA
diastolicBP	1.005	0.992	1.018	0.749	NA
systolicBP	1.003	0.993	1.012	0.559	NA

Standard errors: Robust

Tip

Double adjustment:
Double adjustment in PSM can offer an additional layer of control for confounding and enhance the robustness of treatment effect estimates. The primary goal of dou-

ble adjustment is to further minimize bias in the estimated treatment effect, especially when the propensity score model may not fully balance all covariates across treatment groups. in this approach, The propensity score is estimated, typically using logistic regression, where the treatment assignment is regressed on observed covariates. Units (e.g., individuals) in the treatment group are matched with units in the control group based on their propensity scores. This aims to create comparable groups and balance observed covariates across treatment and control groups. After matching, an *outcome model is fitted to estimate the treatment effect while additionally adjusting for the same covariates used in the propensity score model*. This second adjustment in the outcome model is what constitutes the "double" in "double adjustment."
However, it should be applied thoughtfully, with careful consideration of model specification, covariate selection, and underlying assumptions to ensure valid and reliable results. Always consider the specific context of the study and consult statistical guidelines or experts when applying advanced methods like double adjustment in PSM.

Approach by Austin et al. (2018)

Step 1

```
# Specify the PS model to estimate propensity scores
ps.formula3 <- as.formula(I(bmi == "Not overweight") ~ gender + age + race + income + educatio
                          married + cholesterol + diastolicBP + systolicBP)

# Survey design
require(survey)
analytic.design <- svydesign(id = ~psu, weights = ~interview.weight, strata = ~strata,
                             data = analytic.data, nest = TRUE)
```

Step 2

```r
# Propensity scores
ps.fit3 <- svyglm(ps.formula3, design = analytic.design, family = binomial("logit"))
analytic.data$PS3 <- fitted(ps.fit3)

# Caliper fixing to 0.2*sd(logit of PS)
caliper3 <- 0.2*sd(log(analytic.data$PS3/(1-analytic.data$PS3)))

# Match exposed and unexposed subjects
set.seed(504)
match.obj3 <- matchit(ps.formula, data = analytic.data,
                      distance = analytic.data$PS3,
                      method = "nearest",
                      replace = FALSE,
                      caliper = caliper3,
                      ratio = 1)
analytic.data$PS3 <- match.obj3$distance

# Extract matched data
matched.data3 <- match.data(match.obj3)
```

Step 3

```r
# Balance checking
cov <- c("gender", "age", "race", "income",
         "education", "married", "cholesterol",
         "diastolicBP", "systolicBP")

tab1m3 <- CreateTableOne(strata = "bmi", vars = cov, data = matched.data3, test = F)
print(tab1m3, smd = TRUE)
#>                           Stratified by bmi
#>                            Overweight     Not overweight SMD
#>   n                          2029           2029
#>   gender = Male (%)           917 (45.2)    1009 (49.7)   0.091
#>   age (mean (SD))           47.20 (17.50)  46.26 (17.62)  0.054
#>   race (%)                                                0.071
#>      Black                    443 (21.8)     416 (20.5)
```

```
#>      Hispanic                552 (27.2)     583 (28.7)
#>      Other                   313 (15.4)     351 (17.3)
#>      White                   721 (35.5)     679 (33.5)
#>   income (%)                                            0.067
#>      <25k                    528 (26.0)     506 (24.9)
#>      Between.25kto54k        688 (33.9)     656 (32.3)
#>      Between.55kto99k        439 (21.6)     495 (24.4)
#>      Over100k                374 (18.4)     372 (18.3)
#>   education (%)                                         0.034
#>      College                1165 (57.4)    1186 (58.5)
#>      High.School             654 (32.2)     653 (32.2)
#>      School                  210 (10.3)     190 ( 9.4)
#>   married (%)                                           0.026
#>      Married                1216 (59.9)    1231 (60.7)
#>      Never.married           396 (19.5)     375 (18.5)
#>      Previously.married      417 (20.6)     423 (20.8)
#>   cholesterol (mean (SD)) 176.62 (38.32) 175.13 (38.65)  0.039
#>   diastolicBP (mean (SD))  65.02 (13.44)  64.48 (13.37)  0.041
#>   systolicBP (mean (SD))  117.98 (15.46) 116.37 (17.62)  0.097
```

All SMDs are less than your specified cut-point? If yes, that indicates that there is good covariate balancing.

Step 4

```r
# Setup the design with survey features
analytic.with.miss$matched3 <- 0
analytic.with.miss$matched3[analytic.with.miss$ID %in% matched.data3$ID] <- 1

# Survey setup for full data
w.design03 <- svydesign(strata = ~strata, id = ~psu, weights = ~interview.weight,
                        data = analytic.with.miss, nest = TRUE)

# Subset matched data
w.design.m3 <- subset(w.design03, matched3 == 1)

# Outcome model
out.formula <- as.formula(I(diabetes == "Yes") ~ bmi)
```

```r
fit3 <- svyglm(out.formula, design = w.design.m3, family = binomial("logit"))
summ(fit3, exp = TRUE, confint = TRUE, digits = 3)
```

Observations	4058
Dependent variable	I(diabetes == "Yes")
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R^2 (Cragg-Uhler)	0.071
Pseudo-R^2 (McFadden)	0.054
AIC	2364.398

	exp(Est.)	2.5%	97.5%	t val.	p
(Intercept)	0.169	0.142	0.202	-19.891	0.000
bmiNot overweight	0.244	0.178	0.334	-8.825	0.000

Standard errors: Robust

Double adjustment

```r
# Outcome model with covariates adjustment
fit3.DA <- svyglm(I(diabetes == "Yes") ~ bmi + gender + age + race + income +
                  education + married + cholesterol + diastolicBP + systolicBP,
                design = w.design.m, family = binomial("logit"))
summ(fit3.DA, exp = TRUE, confint = TRUE, digits = 3)
```

Observations	4094
Dependent variable	I(diabetes == "Yes")
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

References

Pseudo-R² (Cragg-Uhler)	0.212
Pseudo-R² (McFadden)	0.166
AIC	2063.466

	exp(Est.)	2.5%	97.5%	t val.	p
(Intercept)	0.045	0.010	0.199	-4.101	NA
bmiNot overweight	0.235	0.159	0.348	-7.219	NA
genderMale	1.317	1.082	1.604	2.740	NA
age	1.051	1.044	1.059	13.280	NA
raceHispanic	0.896	0.649	1.237	-0.665	NA
raceOther	1.158	0.565	2.374	0.401	NA
raceWhite	0.778	0.507	1.193	-1.152	NA
incomeBetween.25kto54k	0.830	0.551	1.251	-0.889	NA
incomeBetween.55kto99k	0.872	0.621	1.224	-0.794	NA
incomeOver100k	0.669	0.418	1.072	-1.669	NA
educationHigh.School	0.948	0.733	1.225	-0.408	NA
educationSchool	0.872	0.448	1.695	-0.405	NA
marriedNever.married	1.055	0.677	1.646	0.237	NA
marriedPreviously.married	0.849	0.534	1.348	-0.694	NA
cholesterol	0.991	0.986	0.995	-3.810	NA
diastolicBP	1.005	0.992	1.018	0.749	NA
systolicBP	1.003	0.993	1.012	0.559	NA

Standard errors: Robust

PSM with MI

The tutorial provides a detailed walkthrough of implementing Propensity Score Matching (PSM) combined with Multiple Imputation (MI) in a statistical analysis, focusing on handling missing data and mitigating bias in observational studies.

The initial chunk is dedicated to loading various R packages that will be utilized throughout the tutorial. These libraries provide functions and tools that facilitate data manipulation, statistical modeling, visualization, and more.

```r
# Load required packages
library(MatchIt)
require(tableone)
require(survey)
require(cobalt)
require(Publish)
require(optmatch)
require(data.table)
require(jtools)
require(ggstance)
require(DataExplorer)
require(mitools)
library(kableExtra)
library(mice)
```

Problem Statement

Logistic regression

- Perform multiple imputation to deal with missing values; with 3 imputed datasets, 5 iterations,

- fit **survey featured logistic regression** in all of the 3 imputed datasets, and
- obtain the pooled OR (adjusted) and the corresponding 95% confidence intervals.
- Hints
  - Use the **covariates** (listed below) in the imputation model.
  - **Imputation model covariates** can be different than the original analysis covariates. You are encouraged to use variables in the imputation model that can be predictive of the variables with missing observations. In this example, we use the strata variable as an **auxiliary variable** in the imputation model, but not the survey weight or PSU variable.
  - Also the **imputation model specification** can be modified. For example, we use **pmm** method for bmi in the imputation model.
  - Remove any subject ID variable from the imputation model, if created in an intermediate step. Indeed ID variables should not be in the imputation model, if they are **not predictive** of the variables with missing observations.

Tip

Predictive Mean Matching:
The "Predictive Mean Matching" (PMM) method in Multiple Imputation (MI) is a widely used technique to handle missing data, particularly well-suited for continuous variables. PMM operates by first creating a predictive model for the variable with missing data, using observed values from other variables in the dataset. For each missing value, PMM identifies a set of observed values with predicted scores that are close to the predicted score for the missing value, derived from the predictive model. Then, instead of imputing a predicted score directly, PMM randomly selects one of the observed values from this set and assigns it as the imputed value. This method retains the

original distribution of the imputed variable since it only uses observed values for imputation, and it also tends to preserve relationships between variables. PMM is particularly advantageous when the normality assumption of the imputed variable is questionable, providing a robust and practical approach to managing missing data in various research contexts.

Propensity score matching (Zanutto, 2006)

- Use the **propensity score matching as per Zanutto E. L. (2006)**'s recommendation in all of the imputed datasets.
- Report the pooled OR estimates (adjusted) and corresponding 95% confidence intervals (adjusted OR).

Data and variables

Analytic data

The analytic dataset is saved as NHANES17.RData.

Variables

We are primarily interested in outcome **diabetes** and exposure **whether born in the US (born)**.

Variables under consideration:

- survey features
  - PSU
  - strata
  - survey weight
- Covariates
  - race
  - age
  - marriage

- education
- gender
- BMI
- systolic blood pressure

Pre-processing

The data is loaded and variables of interest are identified.

```
load(file="Data/propensityscore/NHANES17.RData") # read data
ls()
#> [1] "analytic"           "analytic.with.miss" "has_annotations"
dim(analytic.with.miss)
#> [1] 9254   34
vars <- c("ID", # ID
          "psu", "strata", "weight", # Survey features
          "race", "age", "married","education","gender","bmi","systolicBP", # Covariates
          "born", # Exposure
          "diabetes") # Outcome
```

Subset the dataset

The dataset is then subsetted to retain only the relevant variables, ensuring that subsequent analyses are focused and computationally efficient.

```
dat.with.miss <- analytic.with.miss[,vars]
dim(analytic.with.miss)
#> [1] 9254   34
```

Inspect weights

The weights of the observations are inspected and adjusted to avoid issues in subsequent analyses.

```r
summary(dat.with.miss$weight)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>       0   12347   21060   34671   37562  419763
# weight = 0 would create problem in the analysis
# ad-hoc solution to 0 weight problem
dat.with.miss$weight[dat.with.miss$weight == 0] <- 0.00000001
```

Recode the exposure variable

The exposure variable is recoded for clarity and ease of interpretation in results.

```r
dat.with.miss$born <- car::recode(dat.with.miss$born,
recodes = " 'Born in 50 US states or Washingt' =
'Born in US'; 'Others' = 'Others'; else = NA " )
dat.with.miss$born <- factor(dat.with.miss$born, levels = c("Born in US", "Others"))
```

variable types

Variable types are set, ensuring that each variable is treated appropriately in the analyses.

```r
factor.names <- c("race", "married", "education", "gender", "diabetes")
dat.with.miss[,factor.names] <- lapply(dat.with.miss[,factor.names], factor)
```

Inspect extent of missing data problem

A visualization is generated to explore the extent and pattern of missing data in the dataset, which informs the strategy for handling them.

```r
require(DataExplorer)
plot_missing(dat.with.miss)
```

Note that, **multiple imputation then delete** (MID) approach can be applied if the outcome had some missing values. Due to the small number of missingness, MICE may not impute the outcomes BTW.

> Tip
>
> **Multiple imputation then delete (MID)**:
> MID is a specific approach used in the context of multiple imputation (MI) when dealing with missing outcome data. All missing values, including those in the outcome variable, are imputed to create several complete datasets. In subsequent analyses, the imputed values for the outcome variable are deleted, so that only observed outcome values are analyzed. Each dataset (with observed outcome values and imputed predictor values) is analyzed separately, and results are pooled to provide a single estimate.

Logistic regression

Initialization

The MI process is initialized, setting up the framework for subsequent imputations.

```r
imputation <- mice(data = dat.with.miss, maxit = 0, print = FALSE)
```

Setting imputation model covariates

The predictor matrix is adjusted to specify which variables will be used to predict missing values in the imputation model. Setting strata as auxiliary variable:

```r
pred <- imputation$pred
pred
#>            ID psu strata weight race age married education gender bmi
#> ID          0   1      1      1    1   1       1         1      1   1
#> psu         1   0      1      1    1   1       1         1      1   1
#> strata      1   1      0      1    1   1       1         1      1   1
#> weight      1   1      1      0    1   1       1         1      1   1
#> race        1   1      1      1    0   1       1         1      1   1
#> age         1   1      1      1    1   0       1         1      1   1
#> married     1   1      1      1    1   1       0         1      1   1
#> education   1   1      1      1    1   1       1         0      1   1
#> gender      1   1      1      1    1   1       1         1      0   1
#> bmi         1   1      1      1    1   1       1         1      1   0
#> systolicBP  1   1      1      1    1   1       1         1      1   1
#> born        1   1      1      1    1   1       1         1      1   1
#> diabetes    1   1      1      1    1   1       1         1      1   1
#>            systolicBP born diabetes
#> ID                  1    1        1
#> psu                 1    1        1
#> strata              1    1        1
#> weight              1    1        1
#> race                1    1        1
#> age                 1    1        1
#> married             1    1        1
#> education           1    1        1
#> gender              1    1        1
#> bmi                 1    1        1
#> systolicBP          0    1        1
#> born                1    0        1
#> diabetes            1    1        0
pred[,"ID"] <- pred["ID",] <- 0
```

```r
pred[,"psu"] <- pred["psu",] <- 0
pred[,"weight"] <- pred["weight",] <- 0
pred["strata",] <- 0
pred
#>            ID psu strata weight race age married education gender bmi
#> ID          0   0      0      0    0   0       0         0      0   0
#> psu         0   0      0      0    0   0       0         0      0   0
#> strata      0   0      0      0    0   0       0         0      0   0
#> weight      0   0      0      0    0   0       0         0      0   0
#> race        0   0      1      0    0   1       1         1      1   1
#> age         0   0      1      0    1   0       1         1      1   1
#> married     0   0      1      0    1   1       0         1      1   1
#> education   0   0      1      0    1   1       1         0      1   1
#> gender      0   0      1      0    1   1       1         1      0   1
#> bmi         0   0      1      0    1   1       1         1      1   0
#> systolicBP  0   0      1      0    1   1       1         1      1   1
#> born        0   0      1      0    1   1       1         1      1   1
#> diabetes    0   0      1      0    1   1       1         1      1   1
#>            systolicBP born diabetes
#> ID                  0    0        0
#> psu                 0    0        0
#> strata              0    0        0
#> weight              0    0        0
#> race                1    1        1
#> age                 1    1        1
#> married             1    1        1
#> education           1    1        1
#> gender              1    1        1
#> bmi                 1    1        1
#> systolicBP          0    1        1
#> born                1    0        1
#> diabetes            1    1        0
```

Setting imputation model specification

The method for imputing a particular variable is specified (e.g., using Predictive Mean Matching). Here, we add `pmm` for bmi:

```r
meth <- imputation$meth
meth["bmi"] <- "pmm"
```

Impute incomplete data

Multiple datasets are imputed, each providing a different "guess" at the missing values, based on observed data. We are imputing m = 3 times.

```r
imputation <- mice(data = dat.with.miss,
                   seed = 123,
                   predictorMatrix = pred,
                   method = meth,
                   m = 3,
                   maxit = 5,
                   print = FALSE)
impdata <- mice::complete(imputation, action="long")
impdata$.id <- NULL
m <- 3
set.seed(123)
allImputations <-  imputationList(lapply(1:m,
                                         function(n)
                                           subset(impdata,
                                                  subset=.imp==n)))
str(allImputations)
#> List of 2
#>  $ imputations:List of 3
#>   ..$ :'data.frame': 9254 obs. of  14 variables:
#>   .. ..$ .imp      : int [1:9254] 1 1 1 1 1 1 1 1 1 1 ...
#>   .. ..$ ID        : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 93711 937
#>   .. ..$ psu       : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
#>   .. ..$ strata    : Factor w/ 15 levels "134","135","136",..: 12 10 12 1 5 5 3 1 1 14 ...
#>   .. ..$ weight    : num [1:9254] 8540 42567 8338 8723 7065 ...
#>   .. ..$ race      : Factor w/ 4 levels "Black","Hispanic",..: 3 4 1 3 3 3 1 4 3 2 ...
#>   .. ..$ age       : int [1:9254] 57 46 66 50 23 66 75 49 56 36 ...
#>   .. ..$ married   : Factor w/ 3 levels "Married","Never.married",..: 1 1 3 1 2 1 3 3 1 1 .
#>   .. ..$ education : Factor w/ 3 levels "College","High.School",..: 1 2 2 1 1 3 1 2 1 3 ...
#>   .. ..$ gender    : Factor w/ 2 levels "Female","Male": 1 2 1 2 2 1 1 1 2 2 ...
#>   .. ..$ bmi       : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 31.2 21.3 19.7 ...
```

```
#>   .. ..$ systolicBP: int [1:9254] 108 96 200 112 128 124 120 122 108 112 ...
#>   .. ..$ born      : Factor w/ 2 levels "Born in US","Others": 1 1 1 1 1 2 1 1 2 2 ...
#>   .. ..$ diabetes  : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 1 1 1 ...
#>   ..$ :'data.frame': 9254 obs. of  14 variables:
#>   .. ..$ .imp      : int [1:9254] 2 2 2 2 2 2 2 2 2 2 ...
#>   .. ..$ ID        : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 93711 937
#>   .. ..$ psu       : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
#>   .. ..$ strata    : Factor w/ 15 levels "134","135","136",..: 12 10 12 1 5 5 3 1 1 14 ...
#>   .. ..$ weight    : num [1:9254] 8540 42567 8338 8723 7065 ...
#>   .. ..$ race      : Factor w/ 4 levels "Black","Hispanic",..: 3 4 1 3 3 3 1 4 3 2 ...
#>   .. ..$ age       : int [1:9254] 24 49 66 32 34 66 75 80 56 28 ...
#>   .. ..$ married   : Factor w/ 3 levels "Married","Never.married",..: 2 1 3 2 1 1 3 3 1 2 .
#>   .. ..$ education : Factor w/ 3 levels "College","High.School",..: 1 1 2 1 2 3 1 1 1 1 ...
#>   .. ..$ gender    : Factor w/ 2 levels "Female","Male": 1 2 1 2 2 1 1 1 2 2 ...
#>   .. ..$ bmi       : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 24.9 21.3 19.7 ...
#>   .. ..$ systolicBP: int [1:9254] 102 104 136 112 128 120 120 120 108 112 ...
#>   .. ..$ born      : Factor w/ 2 levels "Born in US","Others": 1 1 1 1 1 2 1 1 2 2 ...
#>   .. ..$ diabetes  : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 1 1 1 ...
#>   ..$ :'data.frame': 9254 obs. of  14 variables:
#>   .. ..$ .imp      : int [1:9254] 3 3 3 3 3 3 3 3 3 3 ...
#>   .. ..$ ID        : int [1:9254] 93703 93704 93705 93706 93707 93708 93709 93710 93711 937
#>   .. ..$ psu       : Factor w/ 2 levels "1","2": 2 1 2 2 1 2 1 1 2 2 ...
#>   .. ..$ strata    : Factor w/ 15 levels "134","135","136",..: 12 10 12 1 5 5 3 1 1 14 ...
#>   .. ..$ weight    : num [1:9254] 8540 42567 8338 8723 7065 ...
#>   .. ..$ race      : Factor w/ 4 levels "Black","Hispanic",..: 3 4 1 3 3 3 1 4 3 2 ...
#>   .. ..$ age       : int [1:9254] 47 71 66 71 45 66 75 37 56 47 ...
#>   .. ..$ married   : Factor w/ 3 levels "Married","Never.married",..: 1 1 3 1 1 1 3 1 1 1 .
#>   .. ..$ education : Factor w/ 3 levels "College","High.School",..: 1 1 2 1 1 3 1 1 1 2 ...
#>   .. ..$ gender    : Factor w/ 2 levels "Female","Male": 1 2 1 2 2 1 1 1 2 2 ...
#>   .. ..$ bmi       : num [1:9254] 17.5 15.7 31.7 21.5 18.1 23.7 38.9 15.9 21.3 19.7 ...
#>   .. ..$ systolicBP: int [1:9254] 100 114 162 112 128 166 120 116 108 112 ...
#>   .. ..$ born      : Factor w/ 2 levels "Born in US","Others": 1 1 1 1 1 2 1 1 2 2 ...
#>   .. ..$ diabetes  : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 2 1 1 1 1 ...
#>  $ call       : language imputationList(lapply(1:m, function(n) subset(impdata, subset = .i
#>  - attr(*, "class")= chr "imputationList"
```

Design

A survey design object is created, ensuring that subsequent analyses appropriately account for the survey design.

```
w.design <- svydesign(ids = ~psu, weights = ~weight, strata = ~strata,
                      data = allImputations, nest = TRUE)
```

Survey data analysis

A logistic regression model is fitted to each imputed dataset.

```
model.formula <- as.formula(I(diabetes == 'Yes') ~
                              born + race + age + married +
                              education + gender + bmi + systolicBP)
fit.from.logistic <- with(w.design, svyglm(model.formula, family = binomial("logit")))
#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!

#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!

#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

Pooled estimates

Results from models across all imputed datasets are pooled to provide a single estimate, accounting for the uncertainty due to missing data.

```
pooled.estimates <- MIcombine(fit.from.logistic)
summary(pooled.estimates, digits = 2, logeffect=TRUE)
#> Multiple imputation results:
#>       with(w.design, svyglm(model.formula, family = binomial("logit")))
#>       MIcombine.default(fit.from.logistic)
#>                           results      se  (lower  upper) missInfo
#> (Intercept)               0.00013 7.5e-05 3.9e-05 0.00041     22 %
#> bornOthers                1.44729 2.7e-01 1.0e+00 2.07384      0 %
#> raceHispanic              0.81619 1.1e-01 6.3e-01 1.05882      0 %
#> raceOther                 1.43817 2.6e-01 1.0e+00 2.04954      3 %
#> raceWhite                 0.86411 1.3e-01 6.5e-01 1.14994      3 %
```

```
#> age                       1.06157 3.6e-03 1.1e+00 1.06874      6 %
#> marriedNever.married      0.83242 1.6e-01 5.7e-01 1.20809     10 %
#> marriedPreviously.married 0.88401 1.1e-01 6.8e-01 1.14163     11 %
#> educationHigh.School      1.16331 1.9e-01 8.4e-01 1.60803      0 %
#> educationSchool           1.41943 2.4e-01 1.0e+00 1.98397      7 %
#> genderMale                1.53458 1.8e-01 1.2e+00 1.94217      3 %
#> bmi                       1.10597 1.2e-02 1.1e+00 1.12956      1 %
#> systolicBP                1.00325 3.2e-03 1.0e+00 1.01001     39 %
OR <- round(exp(pooled.estimates$coefficients), 2)
OR <- as.data.frame(OR)
CI <- round(exp(confint(pooled.estimates)), 2)
OR <- cbind(OR, CI)
OR[2,]
#>              OR 2.5 % 97.5 %
#> bornOthers 1.45  1.01   2.07
```

Propensity score matching analysis

Initialization

The MI process is re-initialized to facilitate PSM in the context of MI.

```
imputation <- mice(data = dat.with.miss, maxit = 0, print = FALSE)
impdata <- mice::complete(imputation, action="long")
m <- 3
allImputations <- imputationList(lapply(1:m,
                                        function(n)
                                          subset(impdata,
                                                 subset=.imp==n)))
```

Zanutto E. L. (2006) under multiple imputation

> **Tip**
>
> An iterative process is performed within each imputed dataset, which involves:
>
> 1. Estimating propensity scores.
> 2. Matching treated and untreated subjects based on these scores.
> 3. Extracting matched data and checking the balance of covariates across matched groups.
> 4. Fitting outcome models to the survey weighted matched data and estimating treatment effects.

Notice that we are performing multi-step process within MI

```r
match.statm <- SMDm <- tab1m <- vector("list", m)
fit.from.PS <- vector("list", m)

for (i in 1:m) {
  analytic.i <- allImputations$imputations[[i]]
  # Rename the weight variable into survey.weight
  names(analytic.i)[names(analytic.i) == "weight"] <- "survey.weight"

  # Specify the PS model to estimate propensity scores
  ps.formula <- as.formula(I(born=="Others") ~
                             race + age + married + education +
                             gender + bmi + systolicBP)

  # Propensity scores
  ps.fit <- glm(ps.formula, data = analytic.i, family = binomial("logit"))
  analytic.i$PS <- fitted(ps.fit)

  # Match exposed and unexposed subjects
  set.seed(123)
  match.obj <- matchit(ps.formula, data = analytic.i,
                       distance = analytic.i$PS,
                       method = "nearest",
                       replace = FALSE,
```

```
                        caliper = 0.2,
                        ratio = 1)
  match.statm[[i]] <- match.obj
  analytic.i$PS <- match.obj$distance

  # Extract matched data
  matched.data <- match.data(match.obj)

  # Balance checking
  cov <- c("race", "age", "married", "education", "gender", "bmi", "systolicBP")

  tab1m[[i]] <- CreateTableOne(strata = "born",
                               vars = cov, data = matched.data,
                               test = FALSE, smd = TRUE)
  SMDm[[i]] <- ExtractSmd(tab1m[[i]])

  # Setup the design with survey features
  analytic.i$matched <- 0
  analytic.i$matched[analytic.i$ID %in% matched.data$ID] <- 1

  # Survey setup for full data
  w.design0 <- svydesign(strata = ~strata, id = ~psu, weights = ~survey.weight,
                         data = analytic.i, nest = TRUE)

  # Subset matched data
  w.design.m <- subset(w.design0, matched == 1)

  # Outcome model (double adjustment)
  out.formula <- as.formula(I(diabetes == "Yes") ~
                              born + race + age + married +
                              education + gender + bmi + systolicBP)
  fit.from.PS[[i]] <- svyglm(out.formula, design = w.design.m,
                     family = quasibinomial("logit"))
}
```

Check matched data

The matched data is inspected to ensure that matching was successful and appropriate.

```r
match.statm
#> [[1]]
#> A matchit object
#>  - method: 1:1 nearest neighbor matching without replacement
#>  - distance: User-defined [caliper]
#>  - caliper: <distance> (0.044)
#>  - number of obs.: 9254 (original), 3590 (matched)
#>  - target estimand: ATT
#>  - covariates: race, age, married, education, gender, bmi, systolicBP
#>
#> [[2]]
#> A matchit object
#>  - method: 1:1 nearest neighbor matching without replacement
#>  - distance: User-defined [caliper]
#>  - caliper: <distance> (0.044)
#>  - number of obs.: 9254 (original), 3598 (matched)
#>  - target estimand: ATT
#>  - covariates: race, age, married, education, gender, bmi, systolicBP
#>
#> [[3]]
#> A matchit object
#>  - method: 1:1 nearest neighbor matching without replacement
#>  - distance: User-defined [caliper]
#>  - caliper: <distance> (0.044)
#>  - number of obs.: 9254 (original), 3594 (matched)
#>  - target estimand: ATT
#>  - covariates: race, age, married, education, gender, bmi, systolicBP
```

Check balance in matched data

The balance of covariates across matched groups is assessed to ensure that matching has successfully reduced bias.

```r
SMDm
#> [[1]]
#>                 1 vs 2
#> race       0.028883793
#> age        0.033614763
#> married    0.007318561
```

```
#> education  0.117536503
#> gender     0.040145831
#> bmi        0.043350560
#> systolicBP 0.054549772
#>
#> [[2]]
#>                 1 vs 2
#> race       0.019901420
#> age        0.016267050
#> married    0.017196043
#> education  0.128811588
#> gender     0.003338016
#> bmi        0.057014434
#> systolicBP 0.071553721
#>
#> [[3]]
#>                1 vs 2
#> race       0.04490482
#> age        0.01959377
#> married    0.03687394
#> education  0.13301810
#> gender     0.01225625
#> bmi        0.03697878
#> systolicBP 0.10025529
```

Pooled estimate

Finally, the treatment effect estimates from the matched analyses across all imputed datasets are pooled to provide a single, overall estimate, ensuring that the final result appropriately accounts for the uncertainty due to both the matching process and the imputation of missing data.

```
pooled.estimates <- MIcombine(fit.from.PS)
summary(pooled.estimates, digits = 2, logeffect=TRUE)
#> Multiple imputation results:
#>       MIcombine.default(fit.from.PS)
#>                           results      se  (lower  upper) missInfo
#> (Intercept)               8.9e-05 4.9e-05 0.00003 0.00026      8 %
```

```
#> bornOthers                2.0e+00 3.1e-01 1.47719 2.73325     16 %
#> raceHispanic              7.0e-01 1.7e-01 0.42593 1.15504     27 %
#> raceOther                 1.4e+00 4.1e-01 0.77209 2.53278     26 %
#> raceWhite                 4.9e-01 2.7e-01 0.15853 1.52308     28 %
#> age                       1.1e+00 4.6e-03 1.04472 1.06298      8 %
#> marriedNever.married      5.9e-01 2.0e-01 0.28824 1.18926     38 %
#> marriedPreviously.married 1.0e+00 2.5e-01 0.62417 1.64438     11 %
#> educationHigh.School      1.4e+00 3.0e-01 0.89403 2.10852      2 %
#> educationSchool           1.3e+00 3.4e-01 0.81042 2.21438      7 %
#> genderMale                1.3e+00 2.3e-01 0.86695 1.83880     31 %
#> bmi                       1.1e+00 1.1e-02 1.08062 1.12285      5 %
#> systolicBP                1.0e+00 3.0e-03 1.00314 1.01494      3 %
OR <- round(exp(pooled.estimates$coefficients), 2)
OR <- as.data.frame(OR)
CI <- round(exp(confint(pooled.estimates)), 2)
OR <- cbind(OR, CI)
OR[2,]
#>              OR 2.5 % 97.5 %
#> bornOthers 2.01  1.48   2.72
```

PS Weighting (US)

Propensity analysis problem

In this chapter, we will use **propensity score weighting** (using SMD cut-point 0.2, may adjust for imbalanced and/or all covariates in the outcome model, if any) analysis as per the following recommendations - (Zanutto 2006) - (DuGoff, Schuler, and Stuart 2014) - (Peter C. Austin, Jembere, and Chiu 2018)

Dataset

- The following modified NHANES dataset
  - NHANES15lab5.RData
- use the data set "analytic.with.miss" within this file.
  - for obtaining the final treatment effect estimates, you can omit missing values, but only after creating the design (e.g., subset the design, not the data itself directly).
- The same dataset was used for **propensity score weighting**

Variables

- Outcome: diabetes
  - 'No' as the reference category
- Exposure: bmi
  - convert to binary with >25 vs. <= 25,
  - with > 25 as the reference category

- Confounder list:
  - gender
  - age
    - assume continuous
  - race
  - income
  - education
  - married
  - cholesterol
  - diastolicBP
  - systolicBP
- Mediator:
  - physical.work
    - 'No' as the reference category
- Survey features
  - psu
  - strata
  - weight

Pre-processing

Load data

```
load(file="Data/propensityscore/NHANES15lab5.RData")
```

Variable summary

```
# Full data
dat.full <- analytic.with.miss

# Exposure
dat.full$bmi <- with(dat.full, ifelse(bmi>25, "Overweight",
                                      ifelse(bmi<=25, "Not overweight", NA)))
dat.full$bmi <- as.factor(dat.full$bmi)
```

```r
dat.full$bmi <- relevel(dat.full$bmi, ref = "Overweight")

# Drop unnecessary variables
dat.full$born <- NULL
dat.full$physical.work <- NULL

# Rename the weight variable into interview.weight
names(dat.full)[names(dat.full) == "weight"] <- "interview.weight"
```

Complete case data

We will use the complete case data to perform the analysis.

```r
# Complete case data
analytic.data <- dat.full[complete.cases(dat.full),]
dim(analytic.data)
#> [1] 6316   15
```

Reproducibility

```r
set.seed(504)
```

Approach by Zanutto (2006)

Step 1

```r
# Specify the PS model to estimate propensity scores
ps.formula <- as.formula(I(bmi=="Not overweight") ~ gender + age + race + income + education +
                           married + cholesterol + diastolicBP + systolicBP)
```

Step 2

For the second step, we will calculate the both unstabilized and stabilized weights. However, stabilized inverse probability

weight is often recommended to prevent from extreme weights (Miguel A. Hernán and Robins 2006).

```
# Propensity scores
ps.fit <- glm(ps.formula, data = analytic.data, family = binomial("logit"))
analytic.data$ps <- predict(ps.fit, type = "response", newdata = analytic.data)

# Unstabilized weight
analytic.data$usweight <- with(analytic.data, ifelse(I(bmi=="Not overweight"),
                                                     1/ps, 1/(1-ps)))

# Unstabilized weight summary
round(summary(analytic.data$usweight), 2)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    1.01    1.33    1.63    2.10    2.14   62.31

# Stabilized weight
analytic.data$sweight <- with(analytic.data, ifelse(I(bmi=="Not overweight"),
                                                    mean(I(bmi=="Not overweight"))/ps,
                                                    (1-mean(I(bmi=="Not overweight")))/(1-ps)))

# Stabilized weight summary
round(summary(analytic.data$sweight), 2)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    0.44    0.70    0.84    1.04    1.08   25.40
```

We can see that the mean of stabilized weights is 1, while it is approximately 2.1 for unstabilized weights. For both unstabilized and stabilized weights, it seems there are extreme weights, particularly for the unstabilized weights. Extreme weights could be dealt with weight truncation, typically truncated at the 1st and 99th percentiles (Cole and Hernán 2008).

Let us truncate the weights at the 1st and 99th percentiles

```
# Truncating unstabilized weight
analytic.data <- analytic.data %>%
  mutate(usweight_t = pmin(pmax(usweight, quantile(usweight, 0.01)),
                           quantile(usweight, 0.99)))
summary(analytic.data$usweight_t)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
#>   1.055   1.335   1.629   2.020   2.140  10.912

# Truncating stabilized weight
analytic.data <- analytic.data %>%
  mutate(sweight_t = pmin(pmax(sweight, quantile(sweight, 0.01)),
                          quantile(sweight, 0.99)))
summary(analytic.data$sweight_t)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.4861  0.7035  0.8400  1.0044  1.0833  4.4901
```

Step 3

Now we will check the distribution of the covariates by the exposure status on the pseudo population in terms of pre-specified SMD.

```
# Covariates
vars <- c("gender", "age", "race", "income", "education", "married", "cholesterol",
         "diastolicBP", "systolicBP")

# Design with truncated unstabilized weight
design.unstab <- svydesign(ids = ~ID, weights = ~usweight_t, data = analytic.data)

# Design with truncated stabilized weight
design.stab <- svydesign(ids = ~ID, weights = ~sweight_t, data = analytic.data)

# Balance checking with truncated unstabilized weight
tab.unstab <- svyCreateTableOne(vars = vars, strata = "bmi", data = design.unstab, test = F)
print(tab.unstab, smd = T)
#>                          Stratified by bmi
#>                           Overweight      Not overweight  SMD
#>   n                        6199.3          6559.7
#>   gender = Male (%)        2999.3 (48.4)   3126.6 (47.7)   0.014
#>   age (mean (SD))           48.38 (17.38)   49.83 (18.11)  0.082
#>   race (%)                                                 0.047
#>      Black                 1304.7 (21.0)   1407.8 (21.5)
#>      Hispanic              1901.9 (30.7)   1873.7 (28.6)
#>      Other                  956.4 (15.4)   1029.3 (15.7)
#>      White                 2036.2 (32.8)   2249.0 (34.3)
```

```
#>   income (%)                                                  0.033
#>      <25k                  1664.3 (26.8)   1829.5 (27.9)
#>      Between.25kto54k      1986.5 (32.0)   2133.5 (32.5)
#>      Between.55kto99k      1449.1 (23.4)   1466.6 (22.4)
#>      Over100k              1099.4 (17.7)   1130.0 (17.2)
#>   education (%)                                               0.019
#>      College               3469.5 (56.0)   3610.4 (55.0)
#>      High.School           2047.7 (33.0)   2217.9 (33.8)
#>      School                 682.1 (11.0)    731.4 (11.1)
#>   married (%)                                                 0.040
#>      Married               3728.2 (60.1)   3853.6 (58.7)
#>      Never.married         1101.9 (17.8)   1147.0 (17.5)
#>      Previously.married    1369.2 (22.1)   1559.1 (23.8)
#>   cholesterol (mean (SD))  181.58 (40.93)  183.13 (43.59)  0.037
#>   diastolicBP (mean (SD))   66.31 (14.64)   66.87 (14.78)  0.038
#>   systolicBP (mean (SD))   121.04 (16.61)  123.60 (23.62)  0.125

# Balance checking with truncated stabilized weight
tab.stab <- svyCreateTableOne(vars = vars, strata = "bmi", data = design.stab, test = F)
print(tab.stab, smd = T)
#>                          Stratified by bmi
#>                           Overweight      Not overweight  SMD
#>   n                        3665.9          2677.9
#>   gender = Male (%)        1771.7 (48.3)   1276.5 (47.7)   0.013
#>   age (mean (SD))           48.40 (17.38)   49.84 (18.12)  0.081
#>   race (%)                                                  0.048
#>      Black                  772.6 (21.1)    575.0 (21.5)
#>      Hispanic              1126.3 (30.7)    764.6 (28.6)
#>      Other                  561.1 (15.3)    420.5 (15.7)
#>      White                 1206.0 (32.9)    917.9 (34.3)
#>   income (%)                                                0.033
#>      <25k                   982.8 (26.8)    747.3 (27.9)
#>      Between.25kto54k      1176.4 (32.1)    870.8 (32.5)
#>      Between.55kto99k       857.6 (23.4)    598.5 (22.3)
#>      Over100k               649.2 (17.7)    461.3 (17.2)
#>   education (%)                                             0.019
#>      College               2051.4 (56.0)   1473.3 (55.0)
#>      High.School           1210.5 (33.0)    905.9 (33.8)
#>      School                 404.0 (11.0)    298.7 (11.2)
#>   married (%)                                               0.040
```

```
#>      Married                2205.9 (60.2)   1572.9 (58.7)
#>      Never.married           650.0 (17.7)    468.0 (17.5)
#>      Previously.married      810.1 (22.1)    637.1 (23.8)
#>   cholesterol (mean (SD))  181.62 (40.91)  183.15 (43.61)  0.036
#>   diastolicBP (mean (SD))   66.37 (14.53)   66.87 (14.81)  0.034
#>   systolicBP (mean (SD))   121.06 (16.58)  123.63 (23.65)  0.126
```

As we can see, all SMDs are less than our specified cut-point of 0.2, indicating that there is good covariate balancing. next, we will fit the outcome model on the pseudo population (i.e., weighted data). Note that we must utilize the survey feature as the design for the population-level estimate. For this step, we will multiply propensity score weight and survey weight and create a `new weight` variable.

Step 4 - with unstabilized weight

```r
require(survey)
require(jtools)

# Create an indicator variable in the full dataset
dat.full$ind <- 0
dat.full$ind[dat.full$ID %in% analytic.data$ID] <- 1

# New weight = interview weight * unstabilized weight
analytic.data$new.usweight_t <- with(analytic.data, interview.weight * usweight_t)
summary(analytic.data$new.usweight_t)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    6437   26567   42862   77768   88106 1831548

#  New weight variable in the full dataset
dat.full$new.usweight_t <- 0
dat.full$new.usweight_t[dat.full$ID %in% analytic.data$ID] <-
  analytic.data$new.usweight_t
summary(dat.full$new.usweight_t)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>       0       0   24068   49261   55161 1831548

```

```r
# Survey setup with full data
w.design0 <- svydesign(id = ~psu, strata = ~strata, weights = ~new.usweight_t,
                       data = dat.full, nest = TRUE)

# Subset the design for analytic sample
w.design.s <- subset(w.design0, ind == 1)

# Outcome model
out.formula <- as.formula(I(diabetes == "Yes") ~ bmi)
fit <- svyglm(out.formula, design = w.design.s, family = binomial("logit"))
summ(fit, exp = TRUE, confint = TRUE, digits = 3)
```

Observations	6316
Dependent variable	I(diabetes == "Yes")
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R^2 (Cragg-Uhler)	0.062
Pseudo-R^2 (McFadden)	0.047
AIC	3371.907

	exp(Est.)	2.5%	97.5%	t val.	p
(Intercept)	0.164	0.141	0.190	-24.213	0.000
bmiNot overweight	0.280	0.217	0.361	-9.814	0.000

Standard errors: Robust

Step 4 - with stabilized weight

Similarly, we can fit the outcome model with stabilized weights.

```r
# Create an indicator variable in the full dataset
dat.full$ind <- 0
dat.full$ind[dat.full$ID %in% analytic.data$ID] <- 1

```

```r
# New weight = interview weight * stabilized weight
analytic.data$new.sweight_t <- with(analytic.data, interview.weight * sweight_t)
summary(analytic.data$new.sweight_t)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    2985   13311   22494   39174   43896  753678

#  New weight variable in the full dataset
dat.full$new.sweight_t <- 0
dat.full$new.sweight_t[dat.full$ID %in% analytic.data$ID] <- analytic.data$new.sweight_t
summary(dat.full$new.sweight_t)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>       0       0   12115   24814   28232  753678

# Survey setup with full data
w.design0 <- svydesign(id = ~psu, strata = ~strata, weights = ~new.sweight_t,
                       data = dat.full, nest = TRUE)

# Subset the design for analytic sample
w.design.s2 <- subset(w.design0, ind == 1)

# Outcome model
out.formula <- as.formula(I(diabetes == "Yes") ~ bmi)
fit.stab <- svyglm(out.formula, design = w.design.s2, family = binomial("logit"))
summ(fit.stab, exp = TRUE, confint = TRUE, digits = 3)
```

Observations	6316
Dependent variable	I(diabetes == "Yes")
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R^2 (Cragg-Uhler)	0.055
Pseudo-R^2 (McFadden)	0.041
AIC	3712.451

	exp(Est.)	2.5%	97.5%	t val.	p
(Intercept)	0.164	0.141	0.190	-24.235	0.000
bmiNot overweight	0.280	0.217	0.361	-9.815	0.000

Standard errors: Robust

Double adjustment

```
library(survey)
# Outcome model with covariates adjustment
fit.DA <- svyglm(I(diabetes == "Yes") ~ bmi + gender + age + race + income +
                 education + married + cholesterol + diastolicBP + systolicBP,
               design = w.design.s2, family = binomial("logit"))
summ(fit.DA, exp = TRUE, confint = TRUE, digits = 3)
```

Observations	6316
Dependent variable	I(diabetes == "Yes")
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R^2 (Cragg-Uhler)	0.194
Pseudo-R^2 (McFadden)	0.149
AIC	3331.238

```

# Log odds ratio with p-values
summary(fit.DA, df.resid = degf(w.design.s2))
#>
#> Call:
#> svyglm(formula = I(diabetes == "Yes") ~ bmi + gender + age +
#>     race + income + education + married + cholesterol + diastolicBP +
#>     systolicBP, design = w.design.s2, family = binomial("logit"))
#>
#> Survey design:
#> subset(w.design0, ind == 1)
#>
```

	exp(Est.)	2.5%	97.5%	t val.	p
(Intercept)	0.045	0.015	0.133	-5.608	NA
bmiNot overweight	0.234	0.175	0.312	-9.826	NA
genderMale	1.402	1.190	1.652	4.038	NA
age	1.050	1.041	1.058	11.793	NA
raceHispanic	0.790	0.600	1.040	-1.677	NA
raceOther	0.849	0.428	1.683	-0.470	NA
raceWhite	0.541	0.371	0.789	-3.192	NA
incomeBetween.25kto54k	0.723	0.478	1.093	-1.540	NA
incomeBetween.55kto99k	0.647	0.411	1.018	-1.883	NA
incomeOver100k	0.476	0.309	0.733	-3.371	NA
educationHigh.School	0.937	0.725	1.211	-0.498	NA
educationSchool	0.930	0.568	1.524	-0.286	NA
marriedNever.married	0.895	0.628	1.275	-0.615	NA
marriedPreviously.married	0.791	0.529	1.184	-1.137	NA
cholesterol	0.993	0.990	0.997	-3.760	NA
diastolicBP	1.007	0.999	1.014	1.744	NA
systolicBP	1.002	0.993	1.012	0.529	NA

Standard errors: Robust

```
#> Coefficients:
#>                            Estimate Std. Error t value Pr(>|t|)
#> (Intercept)               -3.095813   0.552073  -5.608 4.99e-05 ***
#> bmiNot overweight         -1.454514   0.148024  -9.826 6.29e-08 ***
#> genderMale                 0.338112   0.083733   4.038  0.00107 **
#> age                        0.048468   0.004110  11.793 5.48e-09 ***
#> raceHispanic              -0.235285   0.140287  -1.677  0.11422
#> raceOther                 -0.164132   0.349277  -0.470  0.64517
#> raceWhite                 -0.613643   0.192233  -3.192  0.00606 **
#> incomeBetween.25kto54k    -0.324911   0.211048  -1.540  0.14451
#> incomeBetween.55kto99k    -0.435472   0.231288  -1.883  0.07927 .
#> incomeOver100k            -0.742995   0.220399  -3.371  0.00420 **
#> educationHigh.School      -0.065157   0.130721  -0.498  0.62540
#> educationSchool           -0.072034   0.251806  -0.286  0.77874
#> marriedNever.married      -0.110932   0.180496  -0.615  0.54803
#> marriedPreviously.married -0.233875   0.205646  -1.137  0.27327
#> cholesterol               -0.006873   0.001828  -3.760  0.00189 **
#> diastolicBP                0.006728   0.003859   1.744  0.10169
```

```
#> systolicBP                 0.002438   0.004605   0.529  0.60430
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Zero or negative residual df; p-values not defined
#>
#> (Dispersion parameter for binomial family taken to be 0.9452878)
#>
#> Number of Fisher Scoring iterations: 6
```

> **Tip**
>
> **Double adjustment**:
> As explained in the previous chapter, double adjustment should be applied thoughtfully, with careful consideration of model specification, covariate selection, and underlying assumptions to ensure valid and reliable results. Always consider the specific context of the study and consult statistical guidelines or experts when applying advanced methods like double adjustment in propensity score analysis.

Approach by DuGoff et al. (2014)

Step 1

```r
# Specify the PS model to estimate propensity scores
ps.formula2 <- as.formula(I(bmi == "Not overweight") ~ gender + age + race + income + educatio
                          married + cholesterol + diastolicBP + systolicBP +
                          psu + strata + interview.weight)
```

Step 2

```r
# Propensity scores
ps.fit2 <- glm(ps.formula2, data = analytic.data, family = binomial("logit"))
analytic.data$ps2 <- fitted(ps.fit2)
```

```r

# Stabilized weight
analytic.data$sweight.dug <- with(analytic.data, ifelse(I(bmi=="Not overweight"),
                                                        mean(I(bmi=="Not overweight"))/ps2,
                                                        (1-mean(I(bmi=="Not overweight")))/(1-ps2)
summary(analytic.data$sweight.dug)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.4349  0.6942  0.8296  1.0307  1.0859 23.0226

# Truncating stabilized weight
analytic.data <- analytic.data %>%
  mutate(sweight.dug_t = pmin(pmax(sweight.dug, quantile(sweight.dug, 0.01)),
                              quantile(sweight.dug, 0.99)))
summary(analytic.data$sweight.dug_t)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.4802  0.6942  0.8296  1.0019  1.0859  4.4041
```

Step 3

```r
# Balance checking
cov2 <- c("gender", "age", "race", "income", "education", "married", "cholesterol",
         "diastolicBP", "systolicBP")

# Design with truncated stabilized weight
design.stab <- svydesign(ids = ~ID, weights = ~sweight.dug_t, data = analytic.data)

# Balance checking with truncated stabilized weight
tab.stab2 <- svyCreateTableOne(vars = cov2, strata = "bmi", data = design.stab, test = F)
print(tab.stab2, smd = T)
#>                            Stratified by bmi
#>                             Overweight      Not overweight  SMD
#>   n                          3652.9          2675.0
#>   gender = Male (%)          1766.5 (48.4)   1276.8 (47.7)   0.013
#>   age (mean (SD))             48.47 (17.42)   49.80 (17.99)  0.075
#>   race (%)                                                   0.056
#>      Black                    769.1 (21.1)    568.7 (21.3)
#>      Hispanic                1121.5 (30.7)    755.4 (28.2)
#>      Other                    553.0 (15.1)    417.0 (15.6)
#>      White                   1209.4 (33.1)    933.9 (34.9)
```

```
#>   income (%)                                                  0.023
#>      <25k                    982.1 (26.9)    734.0 (27.4)
#>      Between.25kto54k       1172.1 (32.1)    871.1 (32.6)
#>      Between.55kto99k        846.1 (23.2)    597.0 (22.3)
#>      Over100k                652.6 (17.9)    472.9 (17.7)
#>   education (%)                                               0.012
#>      College                2049.1 (56.1)   1484.2 (55.5)
#>      High.School            1202.5 (32.9)    893.7 (33.4)
#>      School                  401.3 (11.0)    297.1 (11.1)
#>   married (%)                                                 0.035
#>      Married                2201.3 (60.3)   1581.2 (59.1)
#>      Never.married           647.4 (17.7)    465.6 (17.4)
#>      Previously.married      804.2 (22.0)    628.2 (23.5)
#>   cholesterol (mean (SD))  181.74 (40.94)  182.97 (43.25)  0.029
#>   diastolicBP (mean (SD))   66.46 (14.42)   66.88 (14.79)  0.029
#>   systolicBP (mean (SD))   121.14 (16.59)  123.39 (23.38)  0.111
```

All SMDs are less than our specified cut-point of 0.2.

Step 4

```r
# Create an indicator variable in the full dataset
dat.full$ind <- 0
dat.full$ind[dat.full$ID %in% analytic.data$ID] <- 1

# New weight = interview weight * stabilized weight
analytic.data$new.sweight.dug_t <- with(analytic.data, interview.weight * sweight.dug_t)
summary(analytic.data$new.sweight.dug_t)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    2996   13343   22260   39436   43914  855145

#  New weight variable in the full dataset
dat.full$new.sweight.dug_t <- 0
dat.full$new.sweight.dug_t[dat.full$ID %in% analytic.data$ID] <- analytic.data$new.sweight.dug
summary(dat.full$new.sweight.dug_t)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>       0       0   12058   24980   28225  855145

```

```r
# Survey setup with full data
w.design0 <- svydesign(id = ~psu, strata = ~strata, weights = ~new.sweight.dug_t,
                       data = dat.full, nest = TRUE)

# Subset the design for analytic sample
w.design.s2 <- subset(w.design0, ind == 1)

# Outcome model
out.formula <- as.formula(I(diabetes == "Yes") ~ bmi)
fit.stab.dug <- svyglm(out.formula, design = w.design.s2, family = binomial("logit"))
summ(fit.stab.dug, exp = TRUE, confint = TRUE, digits = 3)
```

Observations	6316
Dependent variable	I(diabetes == "Yes")
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R^2 (Cragg-Uhler)	0.060
Pseudo-R^2 (McFadden)	0.045
AIC	3518.691

	exp(Est.)	2.5%	97.5%	t val.	p
(Intercept)	0.161	0.140	0.185	-26.027	0.000
bmiNot overweight	0.272	0.203	0.364	-8.727	0.000

Standard errors: Robust

Double adjustment

```r
# Outcome model with covariates adjustment
fit2.DA <- svyglm(I(diabetes == "Yes") ~ bmi + gender + age + race + income +
                  education + married + cholesterol + diastolicBP + systolicBP,
                 design = w.design.s2, family = binomial("logit"))
summ(fit2.DA, exp = TRUE, confint = TRUE, digits = 3)
```

Observations	6316
Dependent variable	I(diabetes == "Yes")
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R² (Cragg-Uhler)	0.196
Pseudo-R² (McFadden)	0.152
AIC	3154.781

	exp(Est.)	2.5%	97.5%	t val.	p
(Intercept)	0.043	0.014	0.131	-5.537	NA
bmiNot overweight	0.235	0.169	0.325	-8.730	NA
genderMale	1.393	1.150	1.688	3.381	NA
age	1.049	1.042	1.056	13.564	NA
raceHispanic	0.813	0.624	1.061	-1.524	NA
raceOther	0.841	0.454	1.558	-0.549	NA
raceWhite	0.535	0.369	0.774	-3.316	NA
incomeBetween.25kto54k	0.720	0.492	1.053	-1.693	NA
incomeBetween.55kto99k	0.659	0.440	0.985	-2.035	NA
incomeOver100k	0.491	0.338	0.713	-3.735	NA
educationHigh.School	0.930	0.753	1.150	-0.667	NA
educationSchool	0.994	0.625	1.580	-0.026	NA
marriedNever.married	0.982	0.671	1.439	-0.092	NA
marriedPreviously.married	0.783	0.534	1.150	-1.245	NA
cholesterol	0.993	0.989	0.996	-3.913	NA
diastolicBP	1.006	0.998	1.013	1.426	NA
systolicBP	1.004	0.995	1.012	0.853	NA

Standard errors: Robust

```r

# Log odds ratio with p-values
summary(fit2.DA, df.resid = degf(w.design.s2))
#>
#> Call:
#> svyglm(formula = I(diabetes == "Yes") ~ bmi + gender + age +
#>     race + income + education + married + cholesterol + diastolicBP +
```

```
#>     systolicBP, design = w.design.s2, family = binomial("logit"))
#>
#> Survey design:
#> subset(w.design0, ind == 1)
#>
#> Coefficients:
#>                            Estimate Std. Error t value Pr(>|t|)
#> (Intercept)               -3.143943   0.567807  -5.537 5.70e-05 ***
#> bmiNot overweight         -1.450047   0.166099  -8.730 2.89e-07 ***
#> genderMale                 0.331561   0.098056   3.381  0.00411 **
#> age                        0.047790   0.003523  13.564 7.97e-10 ***
#> raceHispanic              -0.206657   0.135568  -1.524  0.14822
#> raceOther                 -0.172590   0.314341  -0.549  0.59105
#> raceWhite                 -0.625821   0.188746  -3.316  0.00471 **
#> incomeBetween.25kto54k    -0.328863   0.194223  -1.693  0.11107
#> incomeBetween.55kto99k    -0.417687   0.205223  -2.035  0.05989 .
#> incomeOver100k            -0.711500   0.190509  -3.735  0.00199 **
#> educationHigh.School      -0.072053   0.108040  -0.667  0.51496
#> educationSchool           -0.006136   0.236588  -0.026  0.97965
#> marriedNever.married      -0.017832   0.194660  -0.092  0.92822
#> marriedPreviously.married -0.244058   0.196041  -1.245  0.23226
#> cholesterol               -0.007067   0.001806  -3.913  0.00138 **
#> diastolicBP                0.005516   0.003869   1.426  0.17440
#> systolicBP                 0.003739   0.004381   0.853  0.40682
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Zero or negative residual df; p-values not defined
#>
#> (Dispersion parameter for binomial family taken to be 0.9373219)
#>
#> Number of Fisher Scoring iterations: 6
```

Approach by Austin et al. (2018)

Step 1

```r
# Specify the PS model to estimate propensity scores
ps.formula3 <- as.formula(I(bmi == "Not overweight") ~ gender + age + race + income + educatio
                          married + cholesterol + diastolicBP + systolicBP)

# Survey design
require(survey)
analytic.design <- svydesign(id = ~psu, weights = ~interview.weight, strata = ~strata,
                             data = analytic.data, nest = TRUE)
```

Step 2

```r
# Propensity scores
ps.fit3 <- svyglm(ps.formula3, design = analytic.design, family = binomial("logit"))
analytic.data$ps3 <- fitted(ps.fit3)

# Stabilized weight
analytic.data$sweight.aus <- with(analytic.data, ifelse(I(bmi=="Not overweight"),
                                                        mean(I(bmi=="Not overweight"))/ps3,
                                                        (1-mean(I(bmi=="Not overweight")))/(1-ps3)
summary(analytic.data$sweight.aus)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.4436  0.7138  0.8410  1.0392  1.0645 26.2776

# Truncating stabilized weight
analytic.data <- analytic.data %>%
  mutate(sweight.aus_t = pmin(pmax(sweight.aus, quantile(sweight.aus, 0.01)),
                              quantile(sweight.aus, 0.99)))
summary(analytic.data$sweight.aus_t)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.5198  0.7138  0.8410  1.0071  1.0645  4.8830
```

Step 3

```r
# Balance checking
vars <- c("gender", "age", "race", "income", "education", "married", "cholesterol",
          "diastolicBP", "systolicBP")
```

```r

# Design with truncated stabilized weight
design.stab <- svydesign(ids = ~ID, weights = ~sweight.aus_t, data = analytic.data)

# Balance checking with truncated stabilized weight
tab.stab.aus <- svyCreateTableOne(vars = vars, strata = "bmi", data = design.stab, test = F)
print(tab.stab.aus, smd = T)
#>                            Stratified by bmi
#>                             Overweight      Not overweight  SMD
#>   n                          3434.3          2926.8
#>   gender = Male (%)          1609.8 (46.9)   1458.6 (49.8)   0.059
#>   age (mean (SD))             48.51 (17.32)   49.95 (18.14)  0.081
#>   race (%)                                                   0.099
#>      Black                    737.1 (21.5)    623.8 (21.3)
#>      Hispanic                1085.0 (31.6)    826.0 (28.2)
#>      Other                    471.2 (13.7)    489.7 (16.7)
#>      White                   1141.0 (33.2)    987.3 (33.7)
#>   income (%)                                                 0.041
#>      <25k                     933.7 (27.2)    813.0 (27.8)
#>      Between.25kto54k        1108.2 (32.3)    946.8 (32.3)
#>      Between.55kto99k         776.7 (22.6)    685.4 (23.4)
#>      Over100k                 615.7 (17.9)    481.6 (16.5)
#>   education (%)                                              0.039
#>      College                 1908.2 (55.6)   1601.1 (54.7)
#>      High.School             1129.9 (32.9)   1011.4 (34.6)
#>      School                   396.2 (11.5)    314.3 (10.7)
#>   married (%)                                                0.030
#>      Married                 2063.9 (60.1)   1737.6 (59.4)
#>      Never.married            605.8 (17.6)    501.4 (17.1)
#>      Previously.married       764.6 (22.3)    687.8 (23.5)
#>   cholesterol (mean (SD))  182.67 (41.11)  182.70 (43.38)  0.001
#>   diastolicBP (mean (SD))   66.76 (14.28)   66.85 (14.85)  0.006
#>   systolicBP (mean (SD))   121.39 (16.74)  123.98 (23.72)  0.126
```

All SMDs are less than our specified cut-point of 0.2.

Step 4

```r
# Create an indicator variable in the full dataset
dat.full$ind <- 0
dat.full$ind[dat.full$ID %in% analytic.data$ID] <- 1

# New weight = interview weight * stabilized weight
analytic.data$new.sweight.aus_t <- with(analytic.data, interview.weight * sweight.aus_t)
summary(analytic.data$new.sweight.aus_t)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    3255   13604   22527   38953   43663  819622

#  New weight variable in the full dataset
dat.full$new.sweight.aus_t <- 0
dat.full$new.sweight.aus_t[dat.full$ID %in% analytic.data$ID] <- analytic.data$new.sweight.aus_
summary(dat.full$new.sweight.aus_t)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>       0       0   12356   24675   28169  819622

# Survey setup with full data
w.design0 <- svydesign(id = ~psu, strata = ~strata, weights = ~new.sweight.aus_t,
                       data = dat.full, nest = TRUE)

# Subset the design for analytic sample
w.design.s2 <- subset(w.design0, ind == 1)

# Outcome model
out.formula <- as.formula(I(diabetes == "Yes") ~ bmi)
fit.stab.aus <- svyglm(out.formula, design = w.design.s2, family = binomial("logit"))
summ(fit.stab.aus, exp = TRUE, confint = TRUE, digits = 3)
```

Observations	6316
Dependent variable	I(diabetes == "Yes")
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R² (Cragg-Uhler)	0.055
Pseudo-R² (McFadden)	0.041
AIC	3622.951

	exp(Est.)	2.5%	97.5%	t val.	p
(Intercept)	0.162	0.140	0.187	-24.912	0.000
bmiNot overweight	0.291	0.225	0.377	-9.397	0.000

Standard errors: Robust

Double adjustment

```
# Outcome model with covariates adjustment
fit3.DA <- svyglm(I(diabetes == "Yes") ~ bmi + gender + age + race + income +
                  education + married + cholesterol + diastolicBP + systolicBP,
                design = w.design.s2, family = binomial("logit"))
summ(fit3.DA, exp = TRUE, confint = TRUE, digits = 3)
```

Observations	6316
Dependent variable	I(diabetes == "Yes")
Type	Survey-weighted generalized linear model
Family	binomial
Link	logit

Pseudo-R² (Cragg-Uhler)	0.193
Pseudo-R² (McFadden)	0.149
AIC	3247.730

```

# Log odds ratio with p-values
summary(fit3.DA, df.resid = degf(w.design.s2))
#>
#> Call:
#> svyglm(formula = I(diabetes == "Yes") ~ bmi + gender + age +
#>     race + income + education + married + cholesterol + diastolicBP +
#>     systolicBP, design = w.design.s2, family = binomial("logit"))
#>
```

	exp(Est.)	2.5%	97.5%	t val.	p
(Intercept)	0.045	0.015	0.129	-5.735	NA
bmiNot overweight	0.233	0.175	0.311	-9.886	NA
genderMale	1.416	1.186	1.690	3.850	NA
age	1.049	1.041	1.057	12.066	NA
raceHispanic	0.773	0.585	1.022	-1.808	NA
raceOther	0.867	0.451	1.666	-0.428	NA
raceWhite	0.520	0.353	0.765	-3.316	NA
incomeBetween.25kto54k	0.700	0.461	1.062	-1.677	NA
incomeBetween.55kto99k	0.650	0.407	1.039	-1.801	NA
incomeOver100k	0.467	0.303	0.721	-3.437	NA
educationHigh.School	0.948	0.744	1.209	-0.428	NA
educationSchool	0.976	0.598	1.592	-0.099	NA
marriedNever.married	0.896	0.629	1.276	-0.609	NA
marriedPreviously.married	0.784	0.525	1.169	-1.195	NA
cholesterol	0.993	0.990	0.997	-3.653	NA
diastolicBP	1.005	0.998	1.013	1.412	NA
systolicBP	1.004	0.995	1.012	0.826	NA

Standard errors: Robust

```
#> Survey design:
#> subset(w.design0, ind == 1)
#>
#> Coefficients:
#>                            Estimate Std. Error t value Pr(>|t|)
#> (Intercept)               -3.109181   0.542121  -5.735 3.94e-05 ***
#> bmiNot overweight         -1.456869   0.147369  -9.886 5.81e-08 ***
#> genderMale                 0.347655   0.090292   3.850  0.00157 **
#> age                        0.047457   0.003933  12.066 4.01e-09 ***
#> raceHispanic              -0.257364   0.142343  -1.808  0.09069 .
#> raceOther                 -0.142755   0.333175  -0.428  0.67440
#> raceWhite                 -0.653902   0.197190  -3.316  0.00470 **
#> incomeBetween.25kto54k    -0.357122   0.212996  -1.677  0.11432
#> incomeBetween.55kto99k    -0.430289   0.238964  -1.801  0.09190 .
#> incomeOver100k            -0.761311   0.221498  -3.437  0.00367 **
#> educationHigh.School      -0.052990   0.123816  -0.428  0.67475
#> educationSchool           -0.024734   0.249884  -0.099  0.92246
#> marriedNever.married      -0.109989   0.180532  -0.609  0.55148
```

```
#> marriedPreviously.married -0.243725   0.203990  -1.195  0.25072
#> cholesterol               -0.006605   0.001808  -3.653  0.00235 **
#> diastolicBP                0.005253   0.003721   1.412  0.17842
#> systolicBP                 0.003618   0.004380   0.826  0.42172
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Zero or negative residual df; p-values not defined
#>
#> (Dispersion parameter for binomial family taken to be 0.9416613)
#>
#> Number of Fisher Scoring iterations: 6
```

References

PSM with MI in subset

Problem

In this chapter, we will use **propensity score matching (PSM)** with **multiple imputation**, focusing on specific subpopulations defined by the study's eligibility criteria. We will use PSM as per DuGoff, Schuler, and Stuart (2014) recommendation, with SMD cut-point 0.2 and adjust for imbalanced and/or all covariates in the outcome model, if any.

The modified dataset from NHANES 2017- 2018, which was also used in **missing data** subpopulations chapter, will be used. This example aims to *demonstrate how to do the missing data analysis using multiple imputation with PSM in the context of complex surveys.*

Pre-processing

Load data

Let us import the dataset:

```
load("Data/missingdata/MIexample.RData")
ls()
#> [1] "dat.full"        "has_annotations"
```

Variables

The dataset (`dat.full`) contains 9,254 subjects with 15 variables:

Survey information

- `studyid`: Respondent sequence number
- `survey.weight`: Full sample 2 year interview weight
- `psu`: Masked pseudo PSU
- `strata`: Masked pseudo strata

Outcome variable

- `cvd`: Whether having cardiovascular disease

Exposure variable

- `rheumatoid`: Whether having rheumatoid arthritis

Covariates

- `age`: age in years at screening
- `sex`
- `education`
- `race`: Race/Ethnicity
- `income`: Family income in $
- `bmi`: Body Mass Index in kg/m^2
- `smoking`: Smoking status
- `htn`: Having hypertension
- `diabetes`: Having diabetes

Data pre-processng

```r
# Categorical age
dat.full$age.cat <- with(dat.full, ifelse(age >= 20 & age < 50, "20-49",
                                   ifelse(age >= 50 & age < 65, "50-64", "65+")))
dat.full$age.cat <- factor(dat.full$age.cat, levels = c("20-49", "50-64", "65+"))
table(dat.full$age.cat, useNA = "always")
#>
#> 20-49 50-64   65+  <NA>
#>  2500  1569  5185     0

# Recode rheumatoid to arthritis
dat.full$arthritis <- car::recode(dat.full$rheumatoid, " 'No' = 'No arthritis';
                                        'Yes' = 'Rheumatoid arthritis' ", as.factor = T)
table(dat.full$arthritis, useNA = "always")
#>
```

```
#>          No arthritis Rheumatoid arthritis                <NA>
#>                 3857                  337                5060
```

Subsetting according to eligibility

We will create the analytic dataset with

- adults aged 20 years or more
- without missing values in outcome (cvd) or exposure (rheumatoid arthritis).

```r
# Drop < 20 years
dat.with.miss <- subset(dat.full, age >= 20)

# Frequency for outcome and exposure
table(dat.with.miss$cvd, useNA = "always") # 6 missing
#>
#>   No  Yes <NA>
#> 4872  691    6
table(dat.with.miss$rheumatoid, useNA = "always") # 1375 missing
#>
#>   No  Yes <NA>
#> 3857  337 1375

# Drop missing in outcome and exposure - dataset with missing values only in covariates
dat.analytic <- dat.with.miss[complete.cases(dat.with.miss$cvd),]
dat.analytic <- dat.analytic[complete.cases(dat.analytic$rheumatoid),]
nrow(dat.analytic)
#> [1] 4191
```

We have 4,191 participants in our analytic dataset. The **general strategy of solution** to implement PSM with MI is as follows:

- We will build the imputation model on 4,191 eligible subjects, and
- Apply PSM on each of the imputed datasets, where we will utilize survey features for population-level estimate
- Pool the estimates using Rubin's rule

variable summary

Let us see the summary statistics as we did in the missing data analysis:

```
# Keep only relevant variables
vars <-  c("studyid", "survey.weight", "psu", "strata", "cvd", "arthritis", "age.cat",
           "sex", "education", "race", "income", "bmi", "smoking", "htn", "diabetes")
dat.analytic2 <- dat.analytic[, vars]

# Create Table 1
vars <- c("arthritis", "age.cat", "sex", "education", "race", "income", "bmi", "smoking",
          "htn", "diabetes")
tab1 <- CreateTableOne(vars = vars, strata = "cvd", data = dat.analytic2, includeNA = F,
                       addOverall = T, test = F)
print(tab1, format = "f", showAllLevels = T)
#>                  Stratified by cvd
#>                   level                     Overall      No
#>   n                                          4191         3823
#>   arthritis       No arthritis               3854         3580
#>                   Rheumatoid arthritis        337          243
#>   age.cat         20-49                      2280         2240
#>                   50-64                      1097          979
#>                   65+                         814          604
#>   sex             Male                       2126         1884
#>                   Female                     2065         1939
#>   education       Less than high school       828          728
#>                   High school                2292         2094
#>                   College graduate or above  1063          993
#>   race            White                      1275         1113
#>                   Black                       998          898
#>                   Hispanic                   1015          958
#>                   Others                      903          854
#>   income          less than $20,000           659          557
#>                   $20,000 to $74,999         1967         1796
#>                   $75,000 and Over           1143         1079
#>   bmi (mean (SD))                           29.28 (7.19) 29.20 (7.18)
#>   smoking         Never smoker               2570         2427
#>                   Previous smoker             882          726
#>                   Current smoker              739          670
#>   htn             No                         1424         1380
```

```
#>                    Yes                        2415          2107
#>   diabetes         No                         3622          3396
#>                    Yes                         566           424
#>                   Stratified by cvd
#>                    Yes
#>   n                  368
#>   arthritis          274
#>                       94
#>   age.cat             40
#>                      118
#>                      210
#>   sex                242
#>                      126
#>   education          100
#>                      198
#>                       70
#>   race               162
#>                      100
#>                       57
#>                       49
#>   income             102
#>                      171
#>                       64
#>   bmi (mean (SD)) 30.09 (7.29)
#>   smoking            143
#>                      156
#>                       69
#>   htn                 44
#>                      308
#>   diabetes           226
#>                      142
```

```
# missingness
DataExplorer::plot_missing(dat.analytic2)
```

Dealing with missing values in covariates

Similar to the previous exercise, we will create 5 imputed datasets with 3 iterations, and the predictive mean matching method for bmi and income. We will use the strata variable as an auxiliary variable in the imputation model but not the survey weight or PSU variable.

Step 0: Set up the imputation model

```
# Step 0: Set imputation model
ini <- mice(data = dat.analytic2, maxit = 0, print = FALSE)
pred <- ini$pred

# Use the strata variable as an auxiliary variable in the imputation model
pred["strata",] <- 0

# Do not use survey weight or PSU variable as auxiliary variables
pred[,"studyid"] <- pred["studyid",] <- 0
pred[,"psu"] <- pred["psu",] <- 0
pred[,"survey.weight"] <- pred["survey.weight",] <- 0

# Set imputation method
meth <- ini$meth
meth["bmi"] <- "pmm"
meth["income"] <- "pmm"
```

```r
meth
#>       studyid survey.weight           psu        strata           cvd
#>            ""            ""            ""            ""            ""
#>     arthritis       age.cat           sex     education          race
#>            ""            ""            ""     "polyreg"            ""
#>        income           bmi       smoking           htn      diabetes
#>         "pmm"         "pmm"            ""      "logreg"      "logreg"
```

Step 1: Imputing missing values using mice for eligible subjects

```r
# Step 1: impute the incomplete data
imputation <- mice(data = dat.analytic2,
                   seed = 123,
                   predictorMatrix = pred,
                   method = meth,
                   m = 5,
                   maxit = 3,
                   print = FALSE)
```

Let us save the datasets.

```r
save(dat.full, dat.analytic, dat.analytic2, imputation,
     file = "Data/propensityscore/analytic_imputed.RData")
```

Now we will combine m = 5 datasets and create a stacked dataset. This dataset should contain 5*4,191 = 20,955 rows.

```r
impdata <- mice::complete(imputation, action="long")
dim(impdata)
#> [1] 20955    17
```

```r
#Remove .id variable from the model as it was created in an intermediate step
impdata$.id <- NULL

# Number of subjects
nrow(impdata)
```

```r
#> [1] 20955

# Missing after imputation
DataExplorer::plot_missing(impdata)
```

There is no missing value after imputation. There is an additional variable (`.imp`) in the imputed dataset, which goes from 1 to m = 5, indicating the first to the fifth imputed datasets.

Step 2: PSM steps 1-3 by DuGoff et al. (2014)

Our next step is to use steps 1-3 of the PSM analysis:

- Step 2.1: Fit the PS model by considering survey features as covariates.
- Step 2.2: Match an exposed subject without replacement within the caliper of 0.2 times the standard deviation of the logit of PS.
- Step 2.3: Balance checking using SMD. Consider SMD <0.2 as a good covariate balancing.

0.0.0.0.1 * Step 2.1: PS model specification

```
# Specify the PS model to estimate propensity scores
ps.formula <- as.formula(I(arthritis == "Rheumatoid arthritis") ~ age.cat + sex +
                           education + race + income + bmi + smoking + htn +
                           diabetes + psu + strata + survey.weight)
```

0.0.0.0.2 * Step 2.2: Estimating PS and matching for each imputed dataset

```
# Null vector or list to store values
caliper <- NULL
dat.matched <- match.obj <- list(NULL)

m <- 5 # 5 imputed dataset

# PSM on each of the imputed datasets
for (ii in 1:m) {
  # Imputed dataset
  dat.imputed <- subset(impdata, .imp == ii)

  # Propensity scores
  ps.fit <- glm(ps.formula, data = dat.imputed, family = binomial("logit"))
  dat.imputed$ps <- fitted(ps.fit)

  # Caliper fixing to 0.2*sd(logit of PS)
  caliper[ii] <- 0.2*sd(log(dat.imputed$ps/(1-dat.imputed$ps)))

  # 1:1 PS matching
  set.seed(504)
  match.obj[[ii]] <- matchit(ps.formula, data = dat.imputed,
                        distance = dat.imputed$ps,
                        method = "nearest",
                        replace = FALSE,
                        caliper = caliper[ii],
                        ratio = 1)
  dat.imputed$ps <- match.obj[[ii]]$distance

  # Extract matched data
  dat.matched[[ii]] <- match.data(match.obj[[ii]])
```

```
}
match.obj
#> [[1]]
#> A matchit object
#>  - method: 1:1 nearest neighbor matching without replacement
#>  - distance: User-defined [caliper]
#>  - caliper: <distance> (0.021)
#>  - number of obs.: 4191 (original), 668 (matched)
#>  - target estimand: ATT
#>  - covariates: age.cat, sex, education, race, income, bmi, smoking, htn, diabetes, psu, str
#>
#> [[2]]
#> A matchit object
#>  - method: 1:1 nearest neighbor matching without replacement
#>  - distance: User-defined [caliper]
#>  - caliper: <distance> (0.02)
#>  - number of obs.: 4191 (original), 666 (matched)
#>  - target estimand: ATT
#>  - covariates: age.cat, sex, education, race, income, bmi, smoking, htn, diabetes, psu, str
#>
#> [[3]]
#> A matchit object
#>  - method: 1:1 nearest neighbor matching without replacement
#>  - distance: User-defined [caliper]
#>  - caliper: <distance> (0.021)
#>  - number of obs.: 4191 (original), 672 (matched)
#>  - target estimand: ATT
#>  - covariates: age.cat, sex, education, race, income, bmi, smoking, htn, diabetes, psu, str
#>
#> [[4]]
#> A matchit object
#>  - method: 1:1 nearest neighbor matching without replacement
#>  - distance: User-defined [caliper]
#>  - caliper: <distance> (0.021)
#>  - number of obs.: 4191 (original), 664 (matched)
#>  - target estimand: ATT
#>  - covariates: age.cat, sex, education, race, income, bmi, smoking, htn, diabetes, psu, str
#>
#> [[5]]
#> A matchit object
```

```
#>  - method: 1:1 nearest neighbor matching without replacement
#>  - distance: User-defined [caliper]
#>  - caliper: <distance> (0.021)
#>  - number of obs.: 4191 (original), 662 (matched)
#>  - target estimand: ATT
#>  - covariates: age.cat, sex, education, race, income, bmi, smoking, htn, diabetes, psu, str
```

```
# Dimension of each of the matched dataset
lapply(dat.matched, dim)
#> [[1]]
#> [1] 668  20
#>
#> [[2]]
#> [1] 666  20
#>
#> [[3]]
#> [1] 672  20
#>
#> [[4]]
#> [1] 664  20
#>
#> [[5]]
#> [1] 662  20
```

0.0.0.0.3 * Step 2.3: Balance checking for each imputed dataset

Now we will check balance in terms of SMD on each dataset.

```
tab1m <- list(NULL)
for (ii in 1:m) {
  # Matched data
  dat <- dat.matched[[ii]]

  # Covariates
  vars <- c("age.cat", "sex", "education", "race", "income", "bmi", "smoking",
            "htn", "diabetes")

  # Balance checking
  tab1m[[ii]] <- CreateTableOne(strata = "arthritis", vars = vars, data = dat, test = F)
```

```r
}
print(tab1m, smd = TRUE)
#> [[1]]
#>                                Stratified by arthritis
#>                                 No arthritis  Rheumatoid arthritis SMD
#>   n                               334           334
#>   age.cat (%)                                                        0.018
#>      20-49                         51 (15.3)     52 (15.6)
#>      50-64                        131 (39.2)    133 (39.8)
#>      65+                          152 (45.5)    149 (44.6)
#>   sex = Female (%)                172 (51.5)    178 (53.3)          0.036
#>   education (%)                                                      0.034
#>      Less than high school         80 (24.0)     84 (25.1)
#>      High school                  204 (61.1)    203 (60.8)
#>      College graduate or above     50 (15.0)     47 (14.1)
#>   race (%)                                                           0.049
#>      White                        105 (31.4)    103 (30.8)
#>      Black                        107 (32.0)    112 (33.5)
#>      Hispanic                      67 (20.1)     69 (20.7)
#>      Others                        55 (16.5)     50 (15.0)
#>   income (%)                                                         0.073
#>      less than $20,000             95 (28.4)    100 (29.9)
#>      $20,000 to $74,999           162 (48.5)    167 (50.0)
#>      $75,000 and Over              77 (23.1)     67 (20.1)
#>   bmi (mean (SD))               30.62 (8.16)  30.54 (7.34)          0.011
#>   smoking (%)                                                        0.052
#>      Never smoker                 158 (47.3)    161 (48.2)
#>      Previous smoker              102 (30.5)    106 (31.7)
#>      Current smoker                74 (22.2)     67 (20.1)
#>   htn = Yes (%)                   283 (84.7)    279 (83.5)          0.033
#>   diabetes = Yes (%)              106 (31.7)    100 (29.9)          0.039
#>
#> [[2]]
#>                                Stratified by arthritis
#>                                 No arthritis  Rheumatoid arthritis SMD
#>   n                               333           333
#>   age.cat (%)                                                        0.055
#>      20-49                         50 (15.0)     52 (15.6)
#>      50-64                        142 (42.6)    133 (39.9)
#>      65+                          141 (42.3)    148 (44.4)
```

```
#>   sex = Female (%)              172 (51.7)    176 (52.9)          0.024
#>   education (%)                                                   0.068
#>      Less than high school       92 (27.6)     84 (25.2)
#>      High school                191 (57.4)    202 (60.7)
#>      College graduate or above   50 (15.0)     47 (14.1)
#>   race (%)                                                        0.146
#>      White                       91 (27.3)    104 (31.2)
#>      Black                      100 (30.0)    110 (33.0)
#>      Hispanic                    79 (23.7)     69 (20.7)
#>      Others                      63 (18.9)     50 (15.0)
#>   income (%)                                                      0.131
#>      less than $20,000          115 (34.5)     96 (28.8)
#>      $20,000 to $74,999         166 (49.8)    175 (52.6)
#>      $75,000 and Over            52 (15.6)     62 (18.6)
#>   bmi (mean (SD))             30.50 (7.88)  30.51 (7.46)          0.001
#>   smoking (%)                                                     0.015
#>      Never smoker               160 (48.0)    161 (48.3)
#>      Previous smoker            104 (31.2)    105 (31.5)
#>      Current smoker              69 (20.7)     67 (20.1)
#>   htn = Yes (%)                 288 (86.5)    278 (83.5)          0.084
#>   diabetes = Yes (%)             97 (29.1)     99 (29.7)          0.013
#>
#> [[3]]
#>                               Stratified by arthritis
#>                                No arthritis  Rheumatoid arthritis SMD
#>   n                              336           336
#>   age.cat (%)                                                     0.077
#>      20-49                       46 (13.7)     52 (15.5)
#>      50-64                      145 (43.2)    133 (39.6)
#>      65+                        145 (43.2)    151 (44.9)
#>   sex = Female (%)              176 (52.4)    179 (53.3)          0.018
#>   education (%)                                                   0.076
#>      Less than high school       80 (23.8)     86 (25.6)
#>      High school                215 (64.0)    203 (60.4)
#>      College graduate or above   41 (12.2)     47 (14.0)
#>   race (%)                                                        0.084
#>      White                      114 (33.9)    105 (31.2)
#>      Black                      114 (33.9)    112 (33.3)
#>      Hispanic                    59 (17.6)     69 (20.5)
#>      Others                      49 (14.6)     50 (14.9)
```

```
#>   income (%)                                                    0.118
#>      less than $20,000            87 (25.9)    104 (31.0)
#>      $20,000 to $74,999          183 (54.5)    166 (49.4)
#>      $75,000 and Over             66 (19.6)     66 (19.6)
#>   bmi (mean (SD))              30.56 (7.71)  30.54 (7.53)          0.003
#>   smoking (%)                                                     0.127
#>      Never smoker                180 (53.6)    161 (47.9)
#>      Previous smoker              89 (26.5)    107 (31.8)
#>      Current smoker               67 (19.9)     68 (20.2)
#>   htn = Yes (%)                  280 (83.3)    281 (83.6)          0.008
#>   diabetes = Yes (%)             104 (31.0)    102 (30.4)          0.013
#>
#> [[4]]
#>                               Stratified by arthritis
#>                                No arthritis  Rheumatoid arthritis SMD
#>   n                              332           332
#>   age.cat (%)                                                     0.062
#>      20-49                        48 (14.5)     52 (15.7)
#>      50-64                       127 (38.3)    133 (40.1)
#>      65+                         157 (47.3)    147 (44.3)
#>   sex = Female (%)               174 (52.4)    175 (52.7)          0.006
#>   education (%)                                                   0.059
#>      Less than high school        79 (23.8)     85 (25.6)
#>      High school                 200 (60.2)    200 (60.2)
#>      College graduate or above    53 (16.0)     47 (14.2)
#>   race (%)                                                        0.034
#>      White                       100 (30.1)    105 (31.6)
#>      Black                       111 (33.4)    108 (32.5)
#>      Hispanic                     71 (21.4)     69 (20.8)
#>      Others                       50 (15.1)     50 (15.1)
#>   income (%)                                                      0.067
#>      less than $20,000            93 (28.0)     95 (28.6)
#>      $20,000 to $74,999          165 (49.7)    172 (51.8)
#>      $75,000 and Over             74 (22.3)     65 (19.6)
#>   bmi (mean (SD))              30.06 (7.85)  30.46 (7.49)          0.052
#>   smoking (%)                                                     0.053
#>      Never smoker                166 (50.0)    161 (48.5)
#>      Previous smoker              97 (29.2)    105 (31.6)
#>      Current smoker               69 (20.8)     66 (19.9)
#>   htn = Yes (%)                  281 (84.6)    278 (83.7)          0.025
```

```
#>   diabetes = Yes (%)              93 (28.0)     98 (29.5)          0.033
#>
#> [[5]]
#>                                Stratified by arthritis
#>                                 No arthritis  Rheumatoid arthritis SMD
#>   n                               331           331
#>   age.cat (%)                                                      0.026
#>      20-49                        50 (15.1)     52 (15.7)
#>      50-64                       137 (41.4)    133 (40.2)
#>      65+                         144 (43.5)    146 (44.1)
#>   sex = Female (%)               172 (52.0)    175 (52.9)          0.018
#>   education (%)                                                    0.025
#>      Less than high school        86 (26.0)     84 (25.4)
#>      High school                 196 (59.2)    200 (60.4)
#>      College graduate or above    49 (14.8)     47 (14.2)
#>   race (%)                                                         0.026
#>      White                        99 (29.9)    103 (31.1)
#>      Black                       111 (33.5)    109 (32.9)
#>      Hispanic                     70 (21.1)     69 (20.8)
#>      Others                       51 (15.4)     50 (15.1)
#>   income (%)                                                       0.074
#>      less than $20,000            84 (25.4)     94 (28.4)
#>      $20,000 to $74,999          181 (54.7)    170 (51.4)
#>      $75,000 and Over             66 (19.9)     67 (20.2)
#>   bmi (mean (SD))              30.17 (7.58)  30.49 (7.86)          0.042
#>   smoking (%)                                                      0.027
#>      Never smoker                165 (49.8)    161 (48.6)
#>      Previous smoker             104 (31.4)    105 (31.7)
#>      Current smoker               62 (18.7)     65 (19.6)
#>   htn = Yes (%)                  276 (83.4)    277 (83.7)          0.008
#>   diabetes = Yes (%)             102 (30.8)     97 (29.3)          0.033
```

For each of the datasets, all SMDs are less than our specified cut-point of 0.2.

Step 3: Outcome modelling

Our next step is to fit the outcome model on each of the imputed dataset. Remember, we must utilize survey features to correctly estimate the standard error.

0.0.0.0.1 * 3.1 Preparing dataset for ineligible subjects

Now we will add the **ineligible subjects**(ineligible by study restriction and unmatched) with the matched datasets, so that we can set up the survey design on the full dataset and then subset the design.

Let us subset the data for ineligible subjects:

```r
# Subset for ineligible
dat.ineligible <- list(NULL)

for(ii in 1:m){
  # Matched dataset
  dat <- dat.matched[[ii]]

  # Create an indicator variable in the full dataset
  dat.full$ineligible <- 1
  dat.full$ineligible[dat.full$studyid %in% dat$studyid] <- 0

  # Subset for ineligible
  dat.ineligible[[ii]] <- subset(dat.full, ineligible == 1)

  # Create the .imp variable on each dataset with .imp 1 to m = 5
  dat.ineligible[[ii]]$.imp <- ii
}

# Dimension of each dataset
lapply(dat.ineligible, dim)
#> [[1]]
#> [1] 8586   19
#>
#> [[2]]
#> [1] 8588   19
#>
#> [[3]]
#> [1] 8582   19
#>
#> [[4]]
#> [1] 8590   19
#>
```

```
#> [[5]]
#> [1] 8592   19
```

The next step is combining matched and ineligible datasets. Before merging, we must ensure the variable names are the same.

```r
# Variables in the matched datasets
names(dat.matched[[3]])
#>  [1] ".imp"          "studyid"       "survey.weight" "psu"
#>  [5] "strata"        "cvd"           "arthritis"     "age.cat"
#>  [9] "sex"           "education"     "race"          "income"
#> [13] "bmi"           "smoking"       "htn"           "diabetes"
#> [17] "ps"            "distance"      "weights"       "subclass"

# Variables in the ineligible datasets
names(dat.ineligible[[3]])
#>  [1] "studyid"       "survey.weight" "psu"           "strata"
#>  [5] "cvd"           "rheumatoid"    "age"           "sex"
#>  [9] "education"     "race"          "income"        "bmi"
#> [13] "smoking"       "htn"           "diabetes"      "age.cat"
#> [17] "arthritis"     "ineligible"    ".imp"
```

Four variables (ps, distance, weights, and subclass) are unavailable in our full, analytic, or ineligible datasets but in the matched datasets. We need to create these 4 variables in the ineligible datasets.

```r
dat.ineligible2 <- list(NULL)

for (ii in 1:m) {
  dat <- dat.ineligible[[ii]]

  # Drop the ineligible variable from the dataset
  dat$ineligible <- NULL

  # Create ps, distance, weights, and subclass
  dat$ps <- NA
  dat$distance <- NA
```

```
  dat$weights <- NA
  dat$subclass <- NA

  # Keep only those variables available in the matched dataset
  vars <- names(dat.matched[[1]])
  dat <- dat[,vars]

  # Ineligible datasets in list format
  dat.ineligible2[[ii]] <- dat
}
```

We created ps, distance, weights, and subclass with missing values for the ineligible participants. Note that it doesn't matter whether there are missing covariate values for ineligible. Since we will create the design on the full dataset and subset the design for only eligible (i.e., matched participants), missing covariate values for ineligible will not impact our analysis.

0.0.0.0.2 * 3.2 Combining eligible (matched) and ineligible (unimputed + unmatched) subjects

Now, we will merge matched eligible and unimputed and unmatched ineligible subjects. We should have m = 5 copies of the full dataset with 9,254 subjects on each.

```
dat.full2 <- list(NULL)

for (ii in 1:m) {
  # Eligible
  d1 <- data.frame(dat.matched[[ii]])
  d1$eligible <- 1

  # Ineligible
  d2 <- data.frame(dat.ineligible2[[ii]])
  d2$eligible <- 0

  # Full data
  d3 <- rbind(d1, d2)
  dat.full2[[ii]] <- d3
}
```

```r
lapply(dat.full2, dim)
#> [[1]]
#> [1] 9254   21
#>
#> [[2]]
#> [1] 9254   21
#>
#> [[3]]
#> [1] 9254   21
#>
#> [[4]]
#> [1] 9254   21
#>
#> [[5]]
#> [1] 9254   21

# Stacked dataset
dat.stacked <- rbindlist(dat.full2)
dim(dat.stacked)
#> [1] 46270    21
```

0.0.0.0.3 * 3.3 Prepating Survey design and subpopulation of eligible

The next step is to create the design on the combined dataset.

```r
allImputations <- imputationList(lapply(1:m, function(n) subset(dat.stacked, subset=.imp==n)))

# Design on full data
w.design0 <- svydesign(ids = ~psu,
                       weights = ~survey.weight,
                       strata = ~strata,
                      data = allImputations,
                      nest = TRUE)

# Subset the design
w.design <- subset(w.design0, eligible == 1)
#> Warning in subset.svyimputationList(w.design0, eligible == 1): subset differed
#> between imputations
```

We can see the length of the subsetted design:

```
lapply(w.design$designs, dim)
#> [[1]]
#> [1] 668  21
#>
#> [[2]]
#> [1] 666  21
#>
#> [[3]]
#> [1] 672  21
#>
#> [[4]]
#> [1] 664  21
#>
#> [[5]]
#> [1] 662  21
```

Now we will run the design-adjusted logistic regression on and pool the estimate using Rubin's rule:

0.0.0.0.4 * Step 3.4: Design adjusted regression analysis

```
# Design-adjusted logistic regression
fit <- with(w.design, svyglm(I(cvd == "Yes") ~ arthritis, family = quasibinomial))
res <- exp(as.data.frame(cbind(coef(fit[[1]]),
                               coef(fit[[2]]),
                               coef(fit[[3]]),
                               coef(fit[[4]]),
                               coef(fit[[5]]))))
names(res) <- paste("OR from m =", 1:m)
round(t(res),2)
#>               (Intercept) arthritisRheumatoid arthritis
#> OR from m = 1        0.18                          1.53
#> OR from m = 2        0.24                          1.09
#> OR from m = 3        0.21                          1.28
#> OR from m = 4        0.17                          1.61
#> OR from m = 5        0.12                          2.32
```

0.0.0.0.5 * Step 3.5: Pooling estimates

```
# Pooled estimate
pooled.estimates <- MIcombine(fit)
OR <- round(exp(pooled.estimates$coefficients), 2)
OR <- as.data.frame(OR)
CI <- round(exp(confint(pooled.estimates)), 2)
OR <- cbind(OR, CI)
OR
#>                                 OR 2.5 % 97.5 %
#> (Intercept)                   0.18  0.09   0.36
#> arthritisRheumatoid arthritis 1.52  0.66   3.48
```

0.0.0.0.6 * Double adjustment

```
# Outcome model with covariates adjustment
fit2 <- with(w.design, svyglm(I(cvd == "Yes") ~ arthritis + age.cat + sex + education +
                               race + income + bmi + smoking + htn + diabetes,
                             family = quasibinomial))

# Pooled estimate
pooled.estimates <- MIcombine(fit2)
OR <- round(exp(pooled.estimates$coefficients), 2)
OR <- as.data.frame(OR)
CI <- round(exp(confint(pooled.estimates)), 2)
OR <- cbind(OR, CI)
OR
#>                                     OR 2.5 % 97.5 %
#> (Intercept)                       0.03  0.00   0.94
#> arthritisRheumatoid arthritis     2.07  0.87   4.92
#> age.cat50-64                      3.32  0.73  15.18
#> age.cat65+                        8.89  1.57  50.24
#> sexFemale                         0.61  0.31   1.19
#> educationHigh school              0.86  0.44   1.67
#> educationCollege graduate or above 1.09  0.35   3.38
#> raceBlack                         0.85  0.38   1.90
#> raceHispanic                      0.40  0.15   1.02
#> raceOthers                        0.81  0.26   2.48
#> income$20,000 to $74,999          0.44  0.18   1.07
#> income$75,000 and Over            0.39  0.15   0.99
```

```
#> bmi                                 1.01  0.93   1.08
#> smokingPrevious smoker              1.68  0.66   4.23
#> smokingCurrent smoker               2.00  0.82   4.91
#> htnYes                              1.14  0.35   3.70
#> diabetesYes                         4.04  1.57  10.40
```

References

PSW with MI in subset

Problem

In this chapter, we will use **propensity score (PS) weighting** with **multiple imputation (MI)**, focusing on specific subpopulations defined by the study's eligibility criteria. Similar to the previous chapter on PSM with MI for subpopulation, the modified dataset from NHANES 2017- 2018, will be used.

Load data

Let us import the dataset:

```
rm(list = ls())
load("Data/propensityscore/analytic_imputed.RData")
ls()
#> [1] "dat.analytic"  "dat.analytic2" "dat.full"      "imputation"
```

- `dat.full`: Full dataset of 9,254 subjects
- `dat.analytic` and `dat.analytic2`: Analytic dataset of 4,191 participants with only missing in the covariates. There are no missing values for the exposure or outcomes.
- `imputation`: m = 5 imputed datasets from `dat.analytic2` using MI.

The **general strategy of solution** to implement PS weighting with MI is as follows:

- We will build the imputation model on 4,191 eligible subjects.
- Apply PS weghting on each of the imputed datasets, where we will utilize survey features for population-level estimate
- Pool the estimates using Rubin's rule

Dealing with missing values in covariates

Step 1: Imputing missing values using mice for eligible subjects

We already completed this step in the previous chapter, where we imputed m = 5 datasets using MI.

Now we will combine 5 datasets and create a stacked dataset. This dataset should contain 5*4,191 = 20,955 rows.

```
# Stacked imputed dataset
impdata <- mice::complete(imputation, action="long")
dim(impdata)
#> [1] 20955    17

#Remove .id variable from the model as it was created in an intermediate step
impdata$.id <- NULL

# Missing after imputation
DataExplorer::plot_missing(impdata)
```

Step 2: PS weighting steps 1-3 by DuGoff et al. (2014)

Our next step is to use steps 1-3 of the PS weighting analysis:

- Step 2.1: Fit the PS model by considering survey features as covariates.
- Step 2.2: Calculate PS weights
- Step 2.3: Balance checking using SMD. Consider SMD <0.2 as a good covariate balancing.

0.0.0.0.1 * Step 2.1: PS model specification

```
# Specify the PS model to estimate propensity scores
ps.formula <- as.formula(I(arthritis == "Rheumatoid arthritis") ~ age.cat + sex +
                           education + race + income + bmi + smoking + htn +
                           diabetes + psu + strata + survey.weight)
```

0.0.0.0.2 * Step 2.2: Estimating PS and calculating weights

```
dat.ps <- list(NULL)

m <- 5 # 5 imputed dataset

# PS weighting on each of the imputed datasets
for (ii in 1:m) {
  # Imputed dataset
  dat.imputed <- subset(impdata, .imp == ii)

  # Propensity scores
  ps.fit <- glm(ps.formula, data = dat.imputed, family = binomial("logit"))
  dat.imputed$ps <- fitted(ps.fit)

  # Stabilized weight
  dat.imputed$sweight <- with(dat.imputed,
                              ifelse(I(arthritis == "Rheumatoid arthritis"),
                                     mean(I(arthritis == "Rheumatoid arthritis"))/ps,
                                     (1-mean(I(arthritis == "Rheumatoid arthritis")))/(1-ps)))

  # Dataset
  dat.ps[[ii]] <- dat.imputed
}

```

```r
# Weight summary
purrr::map_df(dat.ps, function(df){summary(df$sweight)})
#> # A tibble: 5 x 6
#>   Min.         `1st Qu.`   Median      Mean         `3rd Qu.`   Max.
#>   <table[1d]> <table[1d]> <table[1d]> <table[1d]> <table[1d]> <table[1d]>
#> 1 0.1062664   0.9335713   0.9497695   0.9955417   1.028579    10.24382
#> 2 0.1104828   0.9337916   0.9496215   0.9957575   1.030514    10.28014
#> 3 0.1051711   0.9334763   0.9497249   0.9950616   1.028141    10.16236
#> 4 0.1090101   0.9336934   0.9492613   0.9947763   1.029242    10.22359
#> 5 0.1068490   0.9338469   0.9494168   0.9955889   1.029760    10.21509
```

```r
# Dimension of each of the imputed dataset
lapply(dat.ps, dim)
#> [[1]]
#> [1] 4191   18
#>
#> [[2]]
#> [1] 4191   18
#>
#> [[3]]
#> [1] 4191   18
#>
#> [[4]]
#> [1] 4191   18
#>
#> [[5]]
#> [1] 4191   18
```

0.0.0.0.3 * Step 2.3: Balance checking for each imputed dataset

Now we will check balance in terms of SMD on each dataset.

```r
tab1m <- list(NULL)
for (ii in 1:m) {
  # PS weighted imputed data
  dat <- dat.ps[[ii]]

  # Covariates
  vars <- c("age.cat", "sex", "education", "race", "income", "bmi", "smoking",
```

```r
            "htn", "diabetes")

  # Design with truncated stabilized weight
  wdesign <- svydesign(ids = ~studyid, weights = ~sweight, data = dat)

  # Balance checking
  tab1m[[ii]] <- svyCreateTableOne(vars = vars, strata = "arthritis", data = wdesign,
                                   test = F)
}
print(tab1m, smd = TRUE)
#> [[1]]
#>                                Stratified by arthritis
#>                                 No arthritis    Rheumatoid arthritis SMD
#>   n                             3856.0          316.3
#>   age.cat (%)                                                          0.077
#>      20-49                      2096.4 (54.4)   159.8 (50.5)
#>      50-64                      1010.5 (26.2)    89.2 (28.2)
#>      65+                         749.1 (19.4)    67.3 (21.3)
#>   sex = Female (%)              1900.0 (49.3)   144.1 (45.6)         0.075
#>   education (%)                                                        0.053
#>      Less than high school       764.6 (19.8)    60.9 (19.3)
#>      High school                2109.7 (54.7)   180.9 (57.2)
#>      College graduate or above   981.8 (25.5)    74.4 (23.5)
#>   race (%)                                                             0.074
#>      White                      1173.6 (30.4)   107.0 (33.8)
#>      Black                       917.9 (23.8)    70.1 (22.2)
#>      Hispanic                    933.5 (24.2)    73.4 (23.2)
#>      Others                      831.0 (21.6)    65.8 (20.8)
#>   income (%)                                                           0.045
#>      less than $20,000           685.9 (17.8)    59.3 (18.7)
#>      $20,000 to $74,999         2015.6 (52.3)   158.2 (50.0)
#>      $75,000 and Over           1154.5 (29.9)    98.8 (31.3)
#>   bmi (mean (SD))               29.34 (7.29)    29.93 (6.90)         0.082
#>   smoking (%)                                                          0.211
#>      Never smoker               2362.5 (61.3)   162.7 (51.4)
#>      Previous smoker             814.5 (21.1)    91.9 (29.1)
#>      Current smoker              679.1 (17.6)    61.6 (19.5)
#>   htn = Yes (%)                 2403.5 (62.3)   211.4 (66.8)         0.094
#>   diabetes = Yes (%)             523.3 (13.6)    50.9 (16.1)         0.071
#>
```

```
#> [[2]]
#>                                Stratified by arthritis
#>                                 No arthritis    Rheumatoid arthritis SMD
#>   n                              3855.7          317.6
#>   age.cat (%)                                                          0.079
#>      20-49                       2096.4 (54.4)   160.2 (50.4)
#>      50-64                       1010.3 (26.2)    89.8 (28.3)
#>      65+                          749.0 (19.4)    67.6 (21.3)
#>   sex = Female (%)               1899.4 (49.3)   144.0 (45.3)         0.079
#>   education (%)                                                        0.050
#>      Less than high school        765.1 (19.8)    61.3 (19.3)
#>      High school                 2113.3 (54.8)   181.6 (57.2)
#>      College graduate or above    977.2 (25.3)    74.7 (23.5)
#>   race (%)                                                             0.076
#>      White                       1173.6 (30.4)   107.8 (33.9)
#>      Black                        917.9 (23.8)    71.0 (22.4)
#>      Hispanic                     933.2 (24.2)    72.0 (22.7)
#>      Others                       831.0 (21.6)    66.8 (21.0)
#>   income (%)                                                           0.039
#>      less than $20,000            688.0 (17.8)    57.8 (18.2)
#>      $20,000 to $74,999          2015.9 (52.3)   160.1 (50.4)
#>      $75,000 and Over            1151.8 (29.9)    99.7 (31.4)
#>   bmi (mean (SD))                 29.33 (7.22)   29.90 (6.95)         0.081
#>   smoking (%)                                                          0.213
#>      Never smoker                2362.4 (61.3)   162.6 (51.2)
#>      Previous smoker              813.9 (21.1)    91.7 (28.9)
#>      Current smoker               679.3 (17.6)    63.3 (19.9)
#>   htn = Yes (%)                  2394.0 (62.1)   213.6 (67.3)         0.108
#>   diabetes = Yes (%)              522.8 (13.6)    50.3 (15.8)         0.065
#>
#> [[3]]
#>                                Stratified by arthritis
#>                                 No arthritis    Rheumatoid arthritis SMD
#>   n                              3856.6          313.7
#>   age.cat (%)                                                          0.088
#>      20-49                       2096.4 (54.4)   156.9 (50.0)
#>      50-64                       1010.1 (26.2)    88.9 (28.3)
#>      65+                          750.1 (19.4)    67.9 (21.6)
#>   sex = Female (%)               1900.1 (49.3)   142.2 (45.3)         0.079
#>   education (%)                                                        0.069
```

```
#>      Less than high school       765.0 (19.8)    57.8 (18.4)
#>      High school                2112.6 (54.8)   182.5 (58.2)
#>      College graduate or above   979.0 (25.4)    73.4 (23.4)
#>   race (%)                                                          0.082
#>      White                      1174.1 (30.4)   106.8 (34.1)
#>      Black                       918.2 (23.8)    69.9 (22.3)
#>      Hispanic                    933.4 (24.2)    69.8 (22.2)
#>      Others                      830.9 (21.5)    67.2 (21.4)
#>   income (%)                                                        0.095
#>      less than $20,000           698.8 (18.1)    63.2 (20.1)
#>      $20,000 to $74,999         2003.2 (51.9)   148.1 (47.2)
#>      $75,000 and Over           1154.6 (29.9)   102.4 (32.7)
#>   bmi (mean (SD))                29.29 (7.21)   29.80 (6.97)         0.072
#>   smoking (%)                                                       0.216
#>      Never smoker               2362.7 (61.3)   160.3 (51.1)
#>      Previous smoker             814.6 (21.1)    91.0 (29.0)
#>      Current smoker              679.2 (17.6)    62.5 (19.9)
#>   htn = Yes (%)                 2398.6 (62.2)   206.3 (65.8)         0.074
#>   diabetes = Yes (%)             523.7 (13.6)    50.9 (16.2)         0.075
#>
#> [[4]]
#>                                Stratified by arthritis
#>                                 No arthritis    Rheumatoid arthritis SMD
#>   n                             3856.2          312.9
#>   age.cat (%)                                                       0.098
#>      20-49                      2096.3 (54.4)   154.7 (49.5)
#>      50-64                      1010.8 (26.2)    91.2 (29.1)
#>      65+                         749.1 (19.4)    67.0 (21.4)
#>   sex = Female (%)              1900.1 (49.3)   144.4 (46.2)         0.063
#>   education (%)                                                     0.064
#>      Less than high school       763.7 (19.8)    58.4 (18.7)
#>      High school                2115.2 (54.9)   181.5 (58.0)
#>      College graduate or above   977.2 (25.3)    72.9 (23.3)
#>   race (%)                                                          0.078
#>      White                      1173.7 (30.4)   106.5 (34.0)
#>      Black                       918.0 (23.8)    70.1 (22.4)
#>      Hispanic                    933.5 (24.2)    71.2 (22.8)
#>      Others                      831.0 (21.5)    65.0 (20.8)
#>   income (%)                                                        0.096
#>      less than $20,000           694.1 (18.0)    63.9 (20.4)
```

```
#>      $20,000 to $74,999         2009.3 (52.1)   148.3 (47.4)
#>      $75,000 and Over           1152.9 (29.9)   100.7 (32.2)
#>   bmi (mean (SD))                29.28 (7.21)   29.86 (7.13)         0.081
#>   smoking (%)                                                        0.209
#>      Never smoker               2362.7 (61.3)   161.3 (51.6)
#>      Previous smoker             813.8 (21.1)    90.7 (29.0)
#>      Current smoker              679.7 (17.6)    60.9 (19.5)
#>   htn = Yes (%)                 2394.5 (62.1)   206.6 (66.0)         0.082
#>   diabetes = Yes (%)             523.0 (13.6)    50.2 (16.0)         0.069
#>
#> [[5]]
#>                                Stratified by arthritis
#>                                 No arthritis    Rheumatoid arthritis SMD
#>   n                             3855.7          316.8
#>   age.cat (%)                                                        0.080
#>      20-49                      2096.4 (54.4)   159.6 (50.4)
#>      50-64                      1010.3 (26.2)    89.8 (28.4)
#>      65+                         749.1 (19.4)    67.4 (21.3)
#>   sex = Female (%)              1899.6 (49.3)   143.9 (45.4)         0.077
#>   education (%)                                                      0.060
#>      Less than high school       767.4 (19.9)    62.1 (19.6)
#>      High school                2110.3 (54.7)   181.8 (57.4)
#>      College graduate or above   978.0 (25.4)    72.9 (23.0)
#>   race (%)                                                           0.074
#>      White                      1173.5 (30.4)   107.1 (33.8)
#>      Black                       917.9 (23.8)    70.1 (22.1)
#>      Hispanic                    933.4 (24.2)    73.1 (23.1)
#>      Others                      830.9 (21.5)    66.5 (21.0)
#>   income (%)                                                         0.047
#>      less than $20,000           685.0 (17.8)    60.6 (19.1)
#>      $20,000 to $74,999         2022.6 (52.5)   159.2 (50.3)
#>      $75,000 and Over           1148.1 (29.8)    97.0 (30.6)
#>   bmi (mean (SD))                29.31 (7.20)   29.77 (7.04)         0.064
#>   smoking (%)                                                        0.206
#>      Never smoker               2362.7 (61.3)   163.1 (51.5)
#>      Previous smoker             813.3 (21.1)    90.0 (28.4)
#>      Current smoker              679.7 (17.6)    63.6 (20.1)
#>   htn = Yes (%)                 2409.4 (62.5)   210.2 (66.3)         0.080
#>   diabetes = Yes (%)             522.5 (13.6)    50.3 (15.9)         0.066
```

For each of the datasets, all SMDs except for smoking are less than our specified cut-point of 0.2. We will adjust our outcome model for smoking.

Step 3: Outcome modelling

Our next step is to fit the outcome model on each of the imputed dataset. Note that, we must utilize survey features to correctly estimate the standard error. For this step, we will multiply PS weight and survey weight and create a `new weight` variable.

0.0.0.0.1 * 3.1 Calculating new weights

```
dat.ps2 <- list(NULL)

for (ii in 1:m) {
  # PS weighted imputed data
  dat <- dat.ps[[ii]]

  # New weight = survey weight * PS weight
  dat$new_weight <- with(dat, survey.weight * sweight)

  dat.ps2[[ii]] <- dat
}
```

0.0.0.0.2 * 3.2 Preparing dataset for ineligible subjects

Now we will add the **ineligible subjects**(ineligible by study restriction) with the PS weighted datasets, so that we can set up the survey design on the full dataset and then subset the design.

Let us subset the data for ineligible subjects:

```
# Subset for ineligible
dat.ineligible <- list(NULL)

for(ii in 1:m){
```

```r
  # Matched dataset
  dat <- dat.ps[[ii]]

  # Create an indicator variable in the full dataset
  dat.full$ineligible <- 1
  dat.full$ineligible[dat.full$studyid %in% dat$studyid] <- 0

  # Subset for ineligible
  dat.ineligible[[ii]] <- subset(dat.full, ineligible == 1)

  # Create the .imp variable on each dataset with .imp 1 to m = 5
  dat.ineligible[[ii]]$.imp <- ii
}

# Dimension of each dataset
lapply(dat.ineligible, dim)
#> [[1]]
#> [1] 5063   19
#>
#> [[2]]
#> [1] 5063   19
#>
#> [[3]]
#> [1] 5063   19
#>
#> [[4]]
#> [1] 5063   19
#>
#> [[5]]
#> [1] 5063   19
```

The next step is combining matched and ineligible datasets. Before merging, we must ensure the variable names are the same.

```r
# Variables in the matched datasets
names(dat.ps2[[3]])
#>  [1] ".imp"          "studyid"       "survey.weight" "psu"
#>  [5] "strata"        "cvd"           "arthritis"     "age.cat"
#>  [9] "sex"           "education"     "race"          "income"
```

```
#> [13] "bmi"           "smoking"       "htn"           "diabetes"
#> [17] "ps"            "sweight"       "new_weight"

# Variables in the ineligible datasets
names(dat.ineligible[[3]])
#>  [1] "studyid"       "survey.weight" "psu"           "strata"
#>  [5] "cvd"           "rheumatoid"    "age"           "sex"
#>  [9] "education"     "race"          "income"        "bmi"
#> [13] "smoking"       "htn"           "diabetes"      "age.cat"
#> [17] "arthritis"     "ineligible"    ".imp"
```

```
dat.ineligible2 <- list(NULL)

for (ii in 1:m) {
  dat <- dat.ineligible[[ii]]

  # Drop the ineligible variable from the dataset
  dat$ineligible <- NULL

  # Create ps and sweight
  dat$ps <- NA
  dat$sweight <- NA
  dat$new_weight <- NA

  # Keep only those variables available in the matched dataset
  vars <- names(dat.ps2[[1]])
  dat <- dat[,vars]

  # Ineligible datasets in list format
  dat.ineligible2[[ii]] <- dat
}
```

0.0.0.0.3 * 3.2 Combining eligible (imputed and PS weighted) and ineligible (unimputed) subjects

Now, we will merge imputed eligible and unimputed ineligible subjects. We should have m = 5 copies of the full dataset with 9,254 subjects on each.

```r
dat.full2 <- list(NULL)

for (ii in 1:m) {
  # Eligible
  d1 <- data.frame(dat.ps2[[ii]])
  d1$eligible <- 1

  # Ineligible
  d2 <- data.frame(dat.ineligible2[[ii]])
  d2$eligible <- 0

  # Full data
  d3 <- rbind(d1, d2)

  #  New weight variable in the full dataset
  d3$new_weight <- 0
  d3$new_weight[d3$studyid %in% d1$studyid] <- d1$new_weight

  # Full data in list format
  dat.full2[[ii]] <- d3
}
lapply(dat.full2, dim)
#> [[1]]
#> [1] 9254   20
#>
#> [[2]]
#> [1] 9254   20
#>
#> [[3]]
#> [1] 9254   20
#>
#> [[4]]
#> [1] 9254   20
#>
#> [[5]]
#> [1] 9254   20

# Stacked dataset
dat.stacked <- rbindlist(dat.full2)
dim(dat.stacked)
```

```
#> [1] 46270    20
```

0.0.0.0.4 * 3.3 Prepating Survey design and subpopulation of eligible

The next step is to create the design on the combined dataset. Make sure to use the `new weight` variable that combines survey weights and PS weights.

```
allImputations <- imputationList(lapply(1:m, function(n) subset(dat.stacked, subset=.imp==n)))

# Design on full data
w.design0 <- svydesign(ids = ~psu,
                       weights = ~new_weight,
                       strata = ~strata,
                      data = allImputations,
                      nest = TRUE)

# Subset the design
w.design <- subset(w.design0, eligible == 1)
```

We can see the length of the subsetted design:

```
lapply(w.design$designs, dim)
#> [[1]]
#> [1] 4191   20
#>
#> [[2]]
#> [1] 4191   20
#>
#> [[3]]
#> [1] 4191   20
#>
#> [[4]]
#> [1] 4191   20
#>
#> [[5]]
#> [1] 4191   20
```

Now we will run the design-adjusted logistic regression, adjusting for smoking since smoking was not balanced in terms of SMD.

0.0.0.0.5 * Step 3.4: Design adjusted regression analysis

```
# Design-adjusted logistic regression
fit <- with(w.design, svyglm(I(cvd == "Yes") ~ arthritis + smoking, family = quasibinomial))
res <- exp(as.data.frame(cbind(coef(fit[[1]]),
                               coef(fit[[2]]),
                               coef(fit[[3]]),
                               coef(fit[[4]]),
                               coef(fit[[5]]))))
names(res) <- paste("OR from m =", 1:m)
round(t(res),2)
#>               (Intercept) arthritisRheumatoid arthritis smokingPrevious smoker
#> OR from m = 1        0.04                          1.21                   2.93
#> OR from m = 2        0.04                          1.24                   2.93
#> OR from m = 3        0.04                          1.23                   2.93
#> OR from m = 4        0.04                          1.22                   2.92
#> OR from m = 5        0.04                          1.22                   2.92
#>               smokingCurrent smoker
#> OR from m = 1                  1.55
#> OR from m = 2                  1.56
#> OR from m = 3                  1.54
#> OR from m = 4                  1.55
#> OR from m = 5                  1.55
```

0.0.0.0.6 * Step 3.5: Pooling estimates

Now, we will pool the estimate using Rubin's rule:

```
# Pooled estimate
pooled.estimates <- MIcombine(fit)
OR <- round(exp(pooled.estimates$coefficients), 2)
OR <- as.data.frame(OR)
CI <- round(exp(confint(pooled.estimates)), 2)
OR <- cbind(OR, CI)
OR
#>                                 OR 2.5 % 97.5 %
```

```
#> (Intercept)                    0.04  0.03   0.05
#> arthritisRheumatoid arthritis 1.22  0.67   2.24
#> smokingPrevious smoker        2.92  2.24   3.82
#> smokingCurrent smoker         1.55  1.14   2.11
```

0.0.0.0.7 * Double adjustment

```
# Outcome model with covariates adjustment
fit2 <- with(w.design, svyglm(I(cvd == "Yes") ~ arthritis + age.cat + sex + education +
                                race + income + bmi + smoking + htn + diabetes,
                              family = quasibinomial))

# Pooled estimate
pooled.estimates <- MIcombine(fit2)
OR <- round(exp(pooled.estimates$coefficients), 2)
OR <- as.data.frame(OR)
CI <- round(exp(confint(pooled.estimates)), 2)
OR <- cbind(OR, CI)
OR
#>                                     OR 2.5 % 97.5 %
#> (Intercept)                       0.02  0.01   0.05
#> arthritisRheumatoid arthritis     1.70  1.07   2.70
#> age.cat50-64                      4.45  2.32   8.51
#> age.cat65+                       11.88  7.13  19.80
#> sexFemale                         0.47  0.35   0.64
#> educationHigh school              0.66  0.36   1.20
#> educationCollege graduate or above 0.63  0.34   1.17
#> raceBlack                         0.97  0.65   1.44
#> raceHispanic                      0.50  0.26   0.98
#> raceOthers                        0.86  0.42   1.78
#> income$20,000 to $74,999          0.49  0.34   0.70
#> income$75,000 and Over            0.31  0.18   0.53
#> bmi                               1.02  0.99   1.06
#> smokingPrevious smoker            1.51  1.17   1.96
#> smokingCurrent smoker             1.47  1.01   2.12
#> htnYes                            1.57  0.84   2.91
#> diabetesYes                       2.91  1.82   4.67
```

References

PSW with multiple tx

Problem

In this chapter, we will use **propensity score weighting (PSW)** for **multiple treatment categories**. We will use CCHS data that was used in the previous chapter on exact matching with CCHS.

Load data

Let us import the dataset:

```
rm(list = ls())
load("Data/propensityscore/cchs123b.RData")
ls()
#> [1] "analytic.miss" "analytic2"
```

- `analytic.miss`: Full dataset of 397,173 participants from CCHS cycles 1.1, 2.1, and 3.1 with missing values in some covariates
- `analytic2`: Analytic dataset of 185,613 participants without missing in the covariates.

Pre-processing

Let us create the full and analytic datasets for only CCHS 3.1.

```
# Full dataset with missing
dat.full <- subset(analytic.miss, cycle == "31")
dim(dat.full)
#> [1] 132221     22
```

```r

# Analytic dataset without missing
dat.analytic <- subset(analytic2, cycle == "31")
dim(dat.analytic)
#> [1] 39634    22
```

We will use the analytic dataset (`dat.analytic`) to run our PSW analysis with the following variables: - Outcome: CVD - Exposure: phyact (3-level physical activity) - Covariates: age, sex, married (marital status), race, edu (education), income, bmi (body mass index), doctor (whether visited to a doctor), stress, smoke, drink (drink alcohol or not), fruit (fruit consumption), bp (blood pressure), diab (diabetes), OA (osteoarthritis), immigrate (immigrant or not)
- Sampling weight: weight

```r
# Is there any character variable?
str(dat.analytic)
#> 'data.frame':    39634 obs. of  22 variables:
#>  $ CVD      : chr  "no event" "no event" "no event" "no event" ...
#>  $ age      : chr  "20-29 years" "65 years and over" "20-29 years" "20-29 years" ...
#>  $ sex      : chr  "Male" "Female" "Male" "Female" ...
#>  $ married  : chr  "not single" "single" "single" "single" ...
#>  $ race     : chr  "White" "White" "White" "White" ...
#>  $ edu      : chr  "2nd grad." "Post-2nd grad." "2nd grad." "Other 2nd grad." ...
#>  $ income   : chr  "$50,000-$79,999" "$29,999 or less" "$29,999 or less" "$50,000-$79,999"
#>  $ bmi      : Factor w/ 3 levels "Underweight",..: 3 2 2 2 2 3 2 3 3 2 ...
#>  $ phyact   : chr  "Inactive" "Moderate" "Active" "Moderate" ...
#>  $ doctor   : chr  "Yes" "Yes" "Yes" "No" ...
#>  $ stress   : chr  "Not too stressed" "Not too stressed" "Not too stressed" "Not too stress
#>  $ smoke    : chr  "Former smoker" "Never smoker" "Never smoker" "Never smoker" ...
#>  $ drink    : chr  "Current drinker" "Former driker" "Never drank" "Current drinker" ...
#>  $ fruit    : Factor w/ 3 levels "0-3 daily serving",..: 2 2 1 1 3 2 2 2 2 3 ...
#>  $ bp       : chr  "No" "No" "No" "No" ...
#>  $ diab     : chr  "No" "No" "No" "No" ...
#>  $ province : chr  "South" "South" "South" "South" ...
#>  $ weight   : num  93.5 111.4 120.4 328.2 810.6 ...
#>  $ cycle    : Factor w/ 3 levels "11","21","31": 3 3 3 3 3 3 3 3 3 3 ...
#>  $ ID       : int  264953 264954 264961 264962 264963 264964 264969 264971 264975 264976 ..
#>  $ OA       : chr  "Control" "OA" "Control" "Control" ...
```

```
#>  $ immigrate: chr  "not immigrant" "not immigrant" "not immigrant" "not immigrant" ...

# Make all variables (except for ID and weight) as factor
var.names <- c("CVD", "phyact", "age", "sex", "married", "race", "edu", "income", "bmi", "docto
               "stress", "smoke", "drink", "fruit", "bp", "diab", "province", "OA", "immigrate"
dat.full[var.names] <- lapply(dat.full[var.names] , factor)
dat.analytic[var.names] <- lapply(dat.analytic[var.names], factor)

# Outcome - CVD
table(dat.analytic$CVD, useNA = "always")
#>
#>    event no event     <NA>
#>     1931    37703        0
dat.full$CVD <- car::recode(dat.full$CVD, "'no event' = 'No'; 'event' = 'Yes'; else = NA ")
dat.full$CVD <- factor(dat.full$CVD, levels = c("No", "Yes"))

dat.analytic$CVD <- car::recode(dat.analytic$CVD, "'no event' = 'No'; 'event' = 'Yes'; else = N
dat.analytic$CVD <- factor(dat.analytic$CVD, levels = c("No", "Yes"))
table(dat.analytic$CVD, useNA = "always")
#>
#>    No   Yes  <NA>
#> 37703  1931     0

# Exposure - physical activity
table(dat.analytic$phyact, useNA = "always")
#>
#>   Active Inactive Moderate     <NA>
#>    11508    17569    10557        0
dat.full$phyact <- factor(dat.full$phyact, levels = c("Inactive", "Moderate", "Active"))
dat.analytic$phyact <- factor(dat.analytic$phyact, levels = c("Inactive", "Moderate", "Active")

# Table 1
vars <- c("age", "sex", "married", "race", "edu", "income", "bmi", "doctor", "stress",
          "smoke", "drink", "fruit", "bp", "diab", "OA", "immigrate")
tab1 <- CreateTableOne(vars = vars, strata = "phyact", data = dat.analytic, test = F)
print(tab1, smd = T, showAllLevels = T)
#>                Stratified by phyact
#>                 level             Inactive      Moderate      Active
#>   n                               17569         10557         11508
#>   age (%)       20-29 years        2537 (14.4)   1709 (16.2)   2316 (20.1)
```

```
#>                 30-39 years        3526 (20.1)   2265 (21.5)   2276 (19.8)
#>                 40-49 years        3270 (18.6)   1939 (18.4)   2037 (17.7)
#>                 50-59 years        2901 (16.5)   1659 (15.7)   1480 (12.9)
#>                 60-64 years        1130 ( 6.4)    665 ( 6.3)    570 ( 5.0)
#>                 65 years and over  3310 (18.8)   1568 (14.9)   1183 (10.3)
#>                 teen                895 ( 5.1)    752 ( 7.1)   1646 (14.3)
#>   sex (%)       Female             9403 (53.5)   5709 (54.1)   5526 (48.0)
#>                 Male               8166 (46.5)   4848 (45.9)   5982 (52.0)
#>   married (%)   not single         9600 (54.6)   5920 (56.1)   5637 (49.0)
#>                 single             7969 (45.4)   4637 (43.9)   5871 (51.0)
#>   race (%)      Non-white          1757 (10.0)    886 ( 8.4)   1066 ( 9.3)
#>                 White             15812 (90.0)   9671 (91.6)  10442 (90.7)
#>   edu (%)       < 2ndary           3690 (21.0)   1635 (15.5)   2039 (17.7)
#>                 2nd grad.          3246 (18.5)   1749 (16.6)   1845 (16.0)
#>                 Other 2nd grad.    1566 ( 8.9)    969 ( 9.2)   1150 (10.0)
#>                 Post-2nd grad.     9067 (51.6)   6204 (58.8)   6474 (56.3)
#>   income (%)    $29,999 or less    4480 (25.5)   1929 (18.3)   1906 (16.6)
#>                 $30,000-$49,999    4018 (22.9)   2059 (19.5)   2097 (18.2)
#>                 $50,000-$79,999    4512 (25.7)   2974 (28.2)   3095 (26.9)
#>                 $80,000 or more    4559 (25.9)   3595 (34.1)   4410 (38.3)
#>   bmi (%)       Underweight         532 ( 3.0)    243 ( 2.3)    340 ( 3.0)
#>                 healthy weight     7349 (41.8)   5023 (47.6)   6233 (54.2)
#>                 Overweight         9688 (55.1)   5291 (50.1)   4935 (42.9)
#>   doctor (%)    No                 2322 (13.2)   1272 (12.0)   1520 (13.2)
#>                 Yes               15247 (86.8)   9285 (88.0)   9988 (86.8)
#>   stress (%)    Not too stressed  13544 (77.1)   8371 (79.3)   9314 (80.9)
#>                 stressed           4025 (22.9)   2186 (20.7)   2194 (19.1)
#>   smoke (%)     Current smoker     5032 (28.6)   2386 (22.6)   2488 (21.6)
#>                 Former smoker      6900 (39.3)   4562 (43.2)   4672 (40.6)
#>                 Never smoker       5637 (32.1)   3609 (34.2)   4348 (37.8)
#>   drink (%)     Current drinker   13913 (79.2)   9016 (85.4)   9863 (85.7)
#>                 Former driker      2582 (14.7)   1102 (10.4)   1063 ( 9.2)
#>                 Never drank        1074 ( 6.1)    439 ( 4.2)    582 ( 5.1)
#>   fruit (%)     0-3 daily serving  5610 (31.9)   2270 (21.5)   1902 (16.5)
#>                 4-6 daily serving  8827 (50.2)   5445 (51.6)   5481 (47.6)
#>                 6+ daily serving   3132 (17.8)   2842 (26.9)   4125 (35.8)
#>   bp (%)        No                14188 (80.8)   8976 (85.0)  10349 (89.9)
#>                 Yes                3381 (19.2)   1581 (15.0)   1159 (10.1)
#>   diab (%)      No                16393 (93.3)  10114 (95.8)  11165 (97.0)
#>                 Yes                1176 ( 6.7)    443 ( 4.2)    343 ( 3.0)
```

```
#>   OA (%)        Control           14864 (84.6)   9310 (88.2)  10565 (91.8)
#>                 OA                 2705 (15.4)   1247 (11.8)    943 ( 8.2)
#>   immigrate (%) not immigrant     16557 (94.2)  10150 (96.1)  11098 (96.4)
#>                 recent             1012 ( 5.8)    407 ( 3.9)    410 ( 3.6)
#>                Stratified by phyact
#>                 SMD
#>   n
#>   age (%)        0.285
#>
#>
#>
#>
#>
#>
#>   sex (%)        0.081
#>
#>   married (%)    0.095
#>
#>   race (%)       0.037
#>
#>   edu (%)        0.119
#>
#>
#>
#>   income (%)     0.213
#>
#>
#>
#>   bmi (%)        0.173
#>
#>
#>   doctor (%)     0.023
#>
#>   stress (%)     0.063
#>
#>   smoke (%)      0.129
#>
#>
#>   drink (%)      0.133
#>
```

```
#>
#>   fruit (%)      0.323
#>
#>
#>   bp (%)         0.175
#>
#>   diab (%)       0.116
#>
#>   OA (%)         0.150
#>
#>   immigrate (%)  0.070
#>
```

PSW for multiple tx

Nominal categories (option 1)

For this part (option 1), we consider physical activity as a nominal variable.

0.0.0.0.1 * Estimating Propensity score

Let us fit the PS model by considering physical activity as a nominal variable and estimate the propensity scores:

```
# Formula
ps.formula <- formula(phyact ~ age + sex + married + race + edu + income + bmi +
                        doctor + stress + smoke + drink + fruit + bp + diab +
                        OA + immigrate)

# PS model
library(VGAM)
#> Warning: package 'VGAM' was built under R version 4.2.3
#> Loading required package: stats4
#> Loading required package: splines
#>
#> Attaching package: 'VGAM'
#> The following object is masked from 'package:survey':
#>
```

```r
#>     calibrate
ps.fit <- vglm(ps.formula, weights = weight, data = dat.analytic,
               family = multinomial(parallel = FALSE))

# Propensity scores
ps <- data.frame(fitted(ps.fit))
head(ps)
#>         Inactive  Moderate    Active
#> 264953 0.3900550 0.2804855 0.3294594
#> 264954 0.5338171 0.2648206 0.2013623
#> 264961 0.5551982 0.1896430 0.2551587
#> 264962 0.4833593 0.2686506 0.2479901
#> 264963 0.1292879 0.1853738 0.6853383
#> 264964 0.4475880 0.2660046 0.2864074

# Summary
apply(ps, 2, summary)
#>           Inactive   Moderate    Active
#> Min.    0.06879258 0.07957314 0.0285961
#> 1st Qu. 0.34233334 0.23371928 0.1867680
#> Median  0.44756388 0.27020295 0.2634537
#> Mean    0.44778510 0.26768926 0.2845256
#> 3rd Qu. 0.55452938 0.30446807 0.3616572
#> Max.    0.89058237 0.39864744 0.7524817
```

0.0.0.0.2 * Creating weights

Let us create PS weights. For subject i, PS weight is calculated as

$$w_i = \frac{1}{P(A_i = a|L)},$$

where A is the exposure with levels a (Inactive, Moderate, and Active in our case), and L is the list of covariates.

```r
# IPW
dat.analytic$ipw <- ifelse(dat.analytic$phyact=="Active", 1/ps$Active,
                           ifelse(dat.analytic$phyact=="Moderate", 1/ps$Moderate,
                                  1/ps$Inactive))
```

```r
with(dat.analytic, by(ipw, phyact, summary))
#> phyact: Inactive
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   1.123   1.671   1.994   2.233   2.505  14.536
#> ------------------------------------------------------------
#> phyact: Moderate
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   2.508   3.206   3.576   3.743   4.087  11.695
#> ------------------------------------------------------------
#> phyact: Active
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   1.329   2.310   3.086   3.534   4.225  25.049
```

0.0.0.0.3 * Balance checking

Now, we will check the balance in terms of SMD:

```r
library(survey)
vars <- c("age", "sex", "married", "race", "edu", "income", "bmi", "doctor", "stress",
          "smoke", "drink", "fruit", "bp", "diab", "OA", "immigrate")

# Design
w.design <- svydesign(id = ~1, weights = ~ipw, data = dat.analytic)

# Table 1
tabw <- svyCreateTableOne(vars = vars, strata = "phyact", data = w.design, test = F)
print(tabw, smd = T, showAllLevels = T)
#>                 Stratified by phyact
#>                  level              Inactive        Moderate
#>   n                                 39231.2         39519.8
#>   age (%)        20-29 years         6477.4 (16.5)   6856.1 (17.3)
#>                  30-39 years         7706.8 (19.6)   8174.8 (20.7)
#>                  40-49 years         7098.2 (18.1)   7163.0 (18.1)
#>                  50-59 years         5922.4 (15.1)   5901.2 (14.9)
#>                  60-64 years         2414.8 ( 6.2)   2265.4 ( 5.7)
#>                  65 years and over   6187.1 (15.8)   5843.8 (14.8)
#>                  teen                3424.6 ( 8.7)   3315.4 ( 8.4)
#>   sex (%)        Female             20188.3 (51.5)  20693.4 (52.4)
#>                  Male               19042.9 (48.5)  18826.3 (47.6)
#>   married (%)    not single         20665.9 (52.7)  20907.0 (52.9)
```

```
#>                 single            18565.3 (47.3)  18612.8 (47.1)
#>   race (%)      Non-white          3471.9 ( 8.8)   4095.5 (10.4)
#>                 White             35759.3 (91.2)  35424.2 (89.6)
#>   edu (%)       < 2ndary           7267.3 (18.5)   7288.2 (18.4)
#>                 2nd grad.          6902.6 (17.6)   6901.3 (17.5)
#>                 Other 2nd grad.    3783.0 ( 9.6)   3671.6 ( 9.3)
#>                 Post-2nd grad.    21278.3 (54.2)  21658.7 (54.8)
#>   income (%)    $29,999 or less    8432.3 (21.5)   8425.7 (21.3)
#>                 $30,000-$49,999    8115.6 (20.7)   8109.8 (20.5)
#>                 $50,000-$79,999   10169.1 (25.9)  10602.5 (26.8)
#>                 $80,000 or more   12514.2 (31.9)  12381.9 (31.3)
#>   bmi (%)       Underweight        1092.7 ( 2.8)   1110.6 ( 2.8)
#>                 healthy weight    18191.1 (46.4)  18613.7 (47.1)
#>                 Overweight        19947.4 (50.8)  19795.5 (50.1)
#>   doctor (%)    No                 5073.1 (12.9)   5127.6 (13.0)
#>                 Yes               34158.1 (87.1)  34392.2 (87.0)
#>   stress (%)    Not too stressed  30921.0 (78.8)  31245.6 (79.1)
#>                 stressed           8310.3 (21.2)   8274.2 (20.9)
#>   smoke (%)     Current smoker    10018.8 (25.5)  10121.1 (25.6)
#>                 Former smoker     15860.1 (40.4)  15737.2 (39.8)
#>                 Never smoker      13352.4 (34.0)  13661.5 (34.6)
#>   drink (%)     Current drinker   32257.4 (82.2)  32836.4 (83.1)
#>                 Former driker      4857.5 (12.4)   4621.5 (11.7)
#>                 Never drank        2116.4 ( 5.4)   2061.9 ( 5.2)
#>   fruit (%)     0-3 daily serving  9738.2 (24.8)   9786.2 (24.8)
#>                 4-6 daily serving 19523.6 (49.8)  19803.8 (50.1)
#>                 6+ daily serving   9969.4 (25.4)   9929.8 (25.1)
#>   bp (%)        No                33045.9 (84.2)  33662.1 (85.2)
#>                 Yes                6185.3 (15.8)   5857.7 (14.8)
#>   diab (%)      No                37227.0 (94.9)  37572.5 (95.1)
#>                 Yes                2004.2 ( 5.1)   1947.3 ( 4.9)
#>   OA (%)        Control           34297.7 (87.4)  34835.2 (88.1)
#>                 OA                 4933.5 (12.6)   4684.6 (11.9)
#>   immigrate (%) not immigrant     37503.8 (95.6)  37532.8 (95.0)
#>                 recent             1727.4 ( 4.4)   1987.0 ( 5.0)
#>                Stratified by phyact
#>                 Active          SMD
#>   n             40667.9
#>   age (%)        6467.5 (15.9)   0.046
#>                  8588.7 (21.1)
```

```
#>                  7538.2 (18.5)
#>                  6327.6 (15.6)
#>                  2420.1 ( 6.0)
#>                  5989.1 (14.7)
#>                  3336.8 ( 8.2)
#>   sex (%)       21026.2 (51.7)   0.012
#>                 19641.7 (48.3)
#>   married (%)   22651.2 (55.7)   0.040
#>                 18016.8 (44.3)
#>   race (%)       3834.0 ( 9.4)   0.034
#>                 36834.0 (90.6)
#>   edu (%)        7395.4 (18.2)   0.023
#>                  6825.8 (16.8)
#>                  3765.2 ( 9.3)
#>                 22681.6 (55.8)
#>   income (%)     8029.8 (19.7)   0.041
#>                  8544.1 (21.0)
#>                 11407.3 (28.0)
#>                 12686.7 (31.2)
#>   bmi (%)        1241.6 ( 3.1)   0.017
#>                 19120.5 (47.0)
#>                 20305.8 (49.9)
#>   doctor (%)     5272.7 (13.0)   0.001
#>                 35395.3 (87.0)
#>   stress (%)    32100.6 (78.9)   0.004
#>                  8567.4 (21.1)
#>   smoke (%)      9760.6 (24.0)   0.032
#>                 17024.0 (41.9)
#>                 13883.4 (34.1)
#>   drink (%)     33927.5 (83.4)   0.022
#>                  4634.1 (11.4)
#>                  2106.3 ( 5.2)
#>   fruit (%)     10170.0 (25.0)   0.007
#>                 20200.6 (49.7)
#>                 10297.3 (25.3)
#>   bp (%)        34372.4 (84.5)   0.017
#>                  6295.6 (15.5)
#>   diab (%)      38850.3 (95.5)   0.020
#>                  1817.7 ( 4.5)
#>   OA (%)        35666.9 (87.7)   0.015
```

```
#>                    5001.1 (12.3)
#>   immigrate (%) 38662.6 (95.1)   0.020
#>                    2005.4 ( 4.9)
```

All covariates are balanced in terms of SMD (SMD $\leq$ 0.20).

0.0.0.0.4 * Outcome analysis

Now, we will fit the outcome model. To get the correct estimate of the standard error, we will set up the design with full data and subset the design.

```
# Create an indicator variable in the full dataset
dat.full$ind <- 0
dat.full$ind[dat.full$ID %in% dat.analytic$ID] <- 1

# New weight = IPW * survey weight
dat.analytic$ATEweight <- with(dat.analytic, ipw * weight)

# New weight variable in the full dataset
dat.full$ATEweight <- 0
dat.full$ATEweight[dat.full$ID %in% dat.analytic$ID] <- dat.analytic$ATEweight

# Survey setup with full data
w.design0 <- svydesign(id = ~1, weights = ~ATEweight, data = dat.full)

# Subset the design for analytic sample
w.design1 <- subset(w.design0, ind == 1)

# Weighted proportion
w.prop <- svyby(formula = ~CVD, by = ~phyact, design = w.design1, FUN = svymean)
w.prop
#>            phyact     CVDNo     CVDYes    se.CVDNo   se.CVDYes
#> Inactive Inactive 0.9665973 0.03340275 0.001503366 0.001503366
#> Moderate Moderate 0.9683972 0.03160284 0.002382517 0.002382517
#> Active     Active 0.9720610 0.02793902 0.002312340 0.002312340

# Outcome model
fit <- svyglm(CVD ~ phyact, design = w.design1, family = binomial)
publish(fit, confint.method = "robust", pvalue.method = "robust")
```

```
#>  Variable    Units OddsRatio       CI.95  p-value
#>    phyact Inactive       Ref
#>           Moderate      0.94 [0.64;1.38]   0.7693
#>             Active      0.83 [0.53;1.29]   0.4133
```

Ordinal categories (for comparison)

For comparison, let us consider physical activity as a ordinal variable (option 2).

0.0.0.0.1 * Define ordinal variable

```
# Exposure - ordinal physical activity
dat.full$phyact.ord <- factor(dat.full$phyact, levels = c("Inactive", "Moderate", "Active"),
                              ordered = T)
dat.analytic$phyact.ord <- factor(dat.analytic$phyact,
                                  levels = c("Inactive", "Moderate", "Active"), ordered = T)
head(dat.analytic$phyact.ord)
#> [1] Inactive Moderate Active   Moderate Active   Active
#> Levels: Inactive < Moderate < Active
```

0.0.0.0.2 * Estimating Propensity score

```
# Formula
ps.formula2 <- formula(phyact.ord ~ age + sex + married + race + edu + income + bmi +
                       doctor + stress + smoke + drink + fruit + bp + diab +
                       OA + immigrate)

# PS model
library(VGAM)
ps.fit2 <- vglm(ps.formula2, weights = weight, data = dat.analytic, family = propodds)

# Propensity scores
ps2 <- data.frame(fitted(ps.fit2))
head(ps2)
#>         Inactive  Moderate    Active
#> 264953 0.3766661 0.2984630 0.3248709
#> 264954 0.5350844 0.2632260 0.2016896
```

```
#> 264961 0.5119910 0.2709983 0.2170107
#> 264962 0.4731775 0.2822563 0.2445662
#> 264963 0.1277460 0.2072136 0.6650404
#> 264964 0.4335105 0.2911417 0.2753478

# Summary
apply(ps2, 2, summary)
#>           Inactive   Moderate     Active
#> Min.    0.07517679 0.07505615 0.03456101
#> 1st Qu. 0.34108922 0.25002048 0.18907940
#> Median  0.44711595 0.28026693 0.26446898
#> Mean    0.44780744 0.26682382 0.28536874
#> 3rd Qu. 0.55497781 0.29473347 0.35967960
#> Max.    0.89038285 0.29934482 0.78152250
```

0.0.0.0.3 * Creating weights

Let us create PS weights:

```
# IPW
dat.analytic$ipw2 <- ifelse(dat.analytic$phyact=="Active", 1/ps2$Active,
                            ifelse(dat.analytic$phyact=="Moderate", 1/ps2$Moderate,
                                   1/ps2$Inactive))
with(dat.analytic, by(ipw2, phyact.ord, summary))
#> phyact.ord: Inactive
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   1.123   1.668   2.001   2.237   2.516  13.302
#> ------------------------------------------------------------
#> phyact.ord: Moderate
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   3.341   3.382   3.525   3.748   3.875  11.248
#> ------------------------------------------------------------
#> phyact.ord: Active
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   1.280   2.319   3.073   3.504   4.211  19.768
```

0.0.0.0.4 * Balance checking

Now, we will check the balance in terms of SMD:

```r
library(survey)
vars <- c("age", "sex", "married", "race", "edu", "income", "bmi", "doctor", "stress",
          "smoke", "drink", "fruit", "bp", "diab", "OA", "immigrate")

# Design
w.design <- svydesign(id = ~1, weights = ~ipw2, data = dat.analytic)

# Table 1
tabw2 <- svyCreateTableOne(vars = vars, strata = "phyact", data = w.design, test = F)
print(tabw2, smd = T, showAllLevels = T)
#>                 Stratified by phyact
#>                  level              Inactive        Moderate
#>   n                                 39305.5         39566.3
#>   age (%)        20-29 years        6679.4 (17.0)   6125.6 (15.5)
#>                  30-39 years        7635.7 (19.4)   8357.8 (21.1)
#>                  40-49 years        7085.9 (18.0)   7124.1 (18.0)
#>                  50-59 years        5820.8 (14.8)   6307.9 (15.9)
#>                  60-64 years        2339.1 ( 6.0)   2494.0 ( 6.3)
#>                  65 years and over  6108.9 (15.5)   6264.9 (15.8)
#>                  teen               3635.8 ( 9.3)   2892.0 ( 7.3)
#>   sex (%)        Female            19987.1 (50.9)  21622.5 (54.6)
#>                  Male              19318.4 (49.1)  17943.8 (45.4)
#>   married (%)    not single        20342.7 (51.8)  22151.4 (56.0)
#>                  single            18962.9 (48.2)  17414.9 (44.0)
#>   race (%)       Non-white          3595.2 ( 9.1)   3507.2 ( 8.9)
#>                  White             35710.4 (90.9)  36059.1 (91.1)
#>   edu (%)        < 2ndary           7467.4 (19.0)   6739.9 (17.0)
#>                  2nd grad.          7027.3 (17.9)   6644.2 (16.8)
#>                  Other 2nd grad.    3815.2 ( 9.7)   3628.7 ( 9.2)
#>                  Post-2nd grad.    20995.6 (53.4)  22553.5 (57.0)
#>   income (%)     $29,999 or less    8596.0 (21.9)   7724.2 (19.5)
#>                  $30,000-$49,999    8182.3 (20.8)   7888.9 (19.9)
#>                  $50,000-$79,999   10126.1 (25.8)  11034.9 (27.9)
#>                  $80,000 or more   12401.1 (31.6)  12918.3 (32.6)
#>   bmi (%)        Underweight        1134.1 ( 2.9)    960.4 ( 2.4)
#>                  healthy weight    18284.0 (46.5)  18426.4 (46.6)
#>                  Overweight        19887.4 (50.6)  20179.5 (51.0)
#>   doctor (%)     No                 5183.0 (13.2)   4741.2 (12.0)
#>                  Yes               34122.5 (86.8)  34825.0 (88.0)
#>   stress (%)     Not too stressed  31032.0 (79.0)  31233.5 (78.9)
```

```
#>                 stressed           8273.6 (21.0)   8332.8 (21.1)
#>   smoke (%)     Current smoker    10263.8 (26.1)   9253.9 (23.4)
#>                 Former smoker     15572.0 (39.6)  16822.9 (42.5)
#>                 Never smoker      13469.8 (34.3)  13489.4 (34.1)
#>   drink (%)     Current drinker   32206.8 (81.9)  33304.9 (84.2)
#>                 Former driker      4898.9 (12.5)   4438.8 (11.2)
#>                 Never drank        2199.8 ( 5.6)   1822.5 ( 4.6)
#>   fruit (%)     0-3 daily serving  9841.8 (25.0)   9509.3 (24.0)
#>                 4-6 daily serving 19586.3 (49.8)  19922.3 (50.4)
#>                 6+ daily serving   9877.5 (25.1)  10134.7 (25.6)
#>   bp (%)        No                33233.3 (84.6)  33115.0 (83.7)
#>                 Yes                6072.2 (15.4)   6451.2 (16.3)
#>   diab (%)      No                37292.2 (94.9)  37648.3 (95.2)
#>                 Yes                2013.4 ( 5.1)   1918.0 ( 4.8)
#>   OA (%)        Control           34458.3 (87.7)  34493.7 (87.2)
#>                 OA                 4847.3 (12.3)   5072.6 (12.8)
#>   immigrate (%) not immigrant     37534.6 (95.5)  37816.7 (95.6)
#>                 recent             1770.9 ( 4.5)   1749.5 ( 4.4)
#>                Stratified by phyact
#>                 Active          SMD
#>   n             40328.5
#>   age (%)        6789.1 (16.8)   0.080
#>                  8504.7 (21.1)
#>                  7562.1 (18.8)
#>                  6079.6 (15.1)
#>                  2283.6 ( 5.7)
#>                  5614.4 (13.9)
#>                  3495.0 ( 8.7)
#>   sex (%)       20379.8 (50.5)   0.055
#>                 19948.8 (49.5)
#>   married (%)   21840.5 (54.2)   0.057
#>                 18488.0 (45.8)
#>   race (%)       4125.6 (10.2)   0.031
#>                 36202.9 (89.8)
#>   edu (%)        7564.8 (18.8)   0.052
#>                  6886.8 (17.1)
#>                  3777.5 ( 9.4)
#>                 22099.4 (54.8)
#>   income (%)     8371.7 (20.8)   0.056
#>                  8581.4 (21.3)
```

```
#>                 11013.7 (27.3)
#>                 12361.7 (30.7)
#>   bmi (%)        1294.7 ( 3.2)   0.037
#>                 19134.4 (47.4)
#>                 19899.5 (49.3)
#>   doctor (%)     5467.9 (13.6)   0.031
#>                 34860.6 (86.4)
#>   stress (%)    31816.9 (78.9)   0.001
#>                  8511.6 (21.1)
#>   smoke (%)     10153.3 (25.2)   0.048
#>                 16300.6 (40.4)
#>                 13874.6 (34.4)
#>   drink (%)     33415.6 (82.9)   0.043
#>                  4714.1 (11.7)
#>                  2198.9 ( 5.5)
#>   fruit (%)     10190.6 (25.3)   0.020
#>                 19953.3 (49.5)
#>                 10184.6 (25.3)
#>   bp (%)        34557.1 (85.7)   0.037
#>                  5771.5 (14.3)
#>   diab (%)      38519.4 (95.5)   0.020
#>                  1809.1 ( 4.5)
#>   OA (%)        35641.0 (88.4)   0.024
#>                  4687.5 (11.6)
#>   immigrate (%) 38153.4 (94.6)   0.030
#>                  2175.2 ( 5.4)
```

Again, all covariates are balanced in terms of SMD.

0.0.0.0.5 * Outcome analysis

Now, we will fit the outcome model:

```r
# Create an indicator variable in the full dataset
dat.full$ind <- 0
dat.full$ind[dat.full$ID %in% dat.analytic$ID] <- 1

# New weight = IPW * survey weight
dat.analytic$ATEweight2 <- with(dat.analytic, ipw2 * weight)

```

```r
# New weight variable in the full dataset
dat.full$ATEweight2 <- 0
dat.full$ATEweight2[dat.full$ID %in% dat.analytic$ID] <- dat.analytic$ATEweight2

# Survey setup with full data
w.design0 <- svydesign(id = ~1, weights = ~ATEweight2, data = dat.full)

# Subset the design for analytic sample
w.design1 <- subset(w.design0, ind == 1)

# Weighted proportion
w.prop2 <- svyby(formula = ~CVD, by = ~phyact, design = w.design1, FUN = svymean)
w.prop2
#>            phyact     CVDNo     CVDYes    se.CVDNo   se.CVDYes
#> Inactive Inactive 0.9670906 0.03290938 0.001472275 0.001472275
#> Moderate Moderate 0.9669294 0.03307061 0.002382617 0.002382617
#> Active     Active 0.9730239 0.02697612 0.002265397 0.002265397

# Outcome model
fit2 <- svyglm(CVD ~ phyact, design = w.design1, family = binomial)
publish(fit2, confint.method = "robust", pvalue.method = "robust")
#>  Variable    Units OddsRatio       CI.95  p-value
#>    phyact Inactive       Ref
#>           Moderate      1.01 [0.71;1.43]   0.9775
#>             Active      0.81 [0.52;1.27]   0.3645
```

Machine learning / GBM (option 3)

In this part, we will use Gradient Boosting as one of the machine learning techniques to estimate the propensity scores.

0.0.0.0.1 * Estimating Propensity score

```r
# Formula
ps.formula3 <- formula(phyact.ord ~ age + sex + married + race + edu + income + bmi +
                       doctor + stress + smoke + drink + fruit + bp + diab +
                       OA + immigrate)

# PS model
```

```r
pacman::p_load(twang)
set.seed(123)
ps.fit3 <- mnps(ps.formula3, data = dat.analytic, estimand = "ATE", verbose = FALSE,
                stop.method = c("es.max"), n.trees = 200, sampw = dat.analytic$weight)
summary(ps.fit3)
#> Summary of pairwise comparisons:
#>   max.std.eff.sz min.p max.ks min.ks.pval stop.method
#> 1      0.4134866     0      1           0         unw
#> 2      0.1880231     0      1           0      es.max
#>
#> Sample sizes and effective sample sizes:
#>   treatment     n ESS:es.max
#> 1  Inactive 17569   7361.137
#> 2  Moderate 10557   4875.807
#> 3    Active 11508   5376.500
```

```r
# Propensity scores
ps3 <- data.frame(Active = ps.fit3$psList$Active$ps,
                  Moderate = ps.fit3$psList$Moderate$ps,
                  Inactive = ps.fit3$psList$Inactive$ps)
names(ps3) <- c("Active","Moderate","Inactive")
head(ps3)
#>      Active  Moderate  Inactive
#> 1 0.3040098 0.2675876 0.4020112
#> 2 0.2418255 0.2600676 0.4910122
#> 3 0.2784937 0.2296143 0.5104120
#> 4 0.2664345 0.2569280 0.4827570
#> 5 0.5330670 0.2407481 0.2759573
#> 6 0.2911926 0.2671368 0.4080280

# Summary
apply(ps3, 2, summary)
#>            Active  Moderate  Inactive
#> Min.    0.1827539 0.1884745 0.2759573
#> 1st Qu. 0.2316300 0.2478217 0.3726945
#> Median  0.2664345 0.2666218 0.4485734
#> Mean    0.2894693 0.2679683 0.4420585
#> 3rd Qu. 0.3343953 0.2859762 0.5015505
#> Max.    0.5330670 0.3255325 0.6389846
```

0.0.0.0.2 * Creating weights

Let us create PS weights:

```
# IPW
dat.analytic$ipw3 <- ifelse(dat.analytic$phyact=="Active", 1/ps3$Active,
                            ifelse(dat.analytic$phyact=="Moderate", 1/ps3$Moderate,
                                   1/ps3$Inactive))
with(dat.analytic, by(ipw3, phyact, summary))
#> phyact: Inactive
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   1.565   1.877   2.086   2.206   2.484   3.624
#> ------------------------------------------------------------
#> phyact: Moderate
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   3.072   3.415   3.652   3.709   3.947   5.306
#> ------------------------------------------------------------
#> phyact: Active
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   1.876   2.716   3.229   3.320   3.952   5.465
```

Weights are not large compared to options 1 and 2.

0.0.0.0.3 * Balance checking

Now, we will check the balance in terms of SMD:

```
library(survey)
vars <- c("age", "sex", "married", "race", "edu", "income", "bmi", "doctor", "stress",
          "smoke", "drink", "fruit", "bp", "diab", "OA", "immigrate")

# Design
w.design <- svydesign(id = ~1, weights = ~ipw3, data = dat.analytic)

# Table 1
tabw <- svyCreateTableOne(vars = vars, strata = "phyact", data = w.design, test = F)
print(tabw, smd = T, showAllLevels = T)
#>                 Stratified by phyact
#>                  level             Inactive        Moderate
#>   n                                38763.2         39159.6
#>   age (%)        20-29 years        6086.5 (15.7)   6641.5 (17.0)
```

```
#>                 30-39 years        7624.6 (19.7)   8171.6 (20.9)
#>                 40-49 years        7057.4 (18.2)   7073.0 (18.1)
#>                 50-59 years        6267.5 (16.2)   5894.9 (15.1)
#>                 60-64 years        2399.6 ( 6.2)   2390.7 ( 6.1)
#>                 65 years and over  6883.8 (17.8)   5859.8 (15.0)
#>                 teen               2443.7 ( 6.3)   3128.1 ( 8.0)
#>   sex (%)       Female            20928.4 (54.0)  21097.6 (53.9)
#>                 Male              17834.8 (46.0)  18062.0 (46.1)
#>   married (%)   not single        21244.6 (54.8)  21279.5 (54.3)
#>                 single            17518.5 (45.2)  17880.1 (45.7)
#>   race (%)      Non-white          3683.4 ( 9.5)   3666.8 ( 9.4)
#>                 White             35079.7 (90.5)  35492.8 (90.6)
#>   edu (%)       < 2ndary           7610.9 (19.6)   6692.3 (17.1)
#>                 2nd grad.          7088.9 (18.3)   6693.3 (17.1)
#>                 Other 2nd grad.    3558.1 ( 9.2)   3619.6 ( 9.2)
#>                 Post-2nd grad.    20505.3 (52.9)  22154.2 (56.6)
#>   income (%)    $29,999 or less    9138.3 (23.6)   7750.6 (19.8)
#>                 $30,000-$49,999    8400.8 (21.7)   7956.3 (20.3)
#>                 $50,000-$79,999    9946.0 (25.7)  10696.8 (27.3)
#>                 $80,000 or more   11278.0 (29.1)  12755.9 (32.6)
#>   bmi (%)       Underweight        1195.7 ( 3.1)    967.7 ( 2.5)
#>                 healthy weight    16766.4 (43.3)  18791.6 (48.0)
#>                 Overweight        20801.0 (53.7)  19400.2 (49.5)
#>   doctor (%)    No                 5070.5 (13.1)   4827.3 (12.3)
#>                 Yes               33692.7 (86.9)  34332.3 (87.7)
#>   stress (%)    Not too stressed  29885.7 (77.1)  31099.5 (79.4)
#>                 stressed           8877.5 (22.9)   8060.0 (20.6)
#>   smoke (%)     Current smoker    10737.3 (27.7)   9401.4 (24.0)
#>                 Former smoker     15204.3 (39.2)  16211.3 (41.4)
#>                 Never smoker      12821.5 (33.1)  13546.9 (34.6)
#>   drink (%)     Current drinker   31123.7 (80.3)  33043.2 (84.4)
#>                 Former driker      5322.0 (13.7)   4301.7 (11.0)
#>                 Never drank        2317.5 ( 6.0)   1814.6 ( 4.6)
#>   fruit (%)     0-3 daily serving 10529.0 (27.2)   8996.8 (23.0)
#>                 4-6 daily serving 19466.5 (50.2)  19851.5 (50.7)
#>                 6+ daily serving   8767.6 (22.6)  10311.3 (26.3)
#>   bp (%)        No                31712.5 (81.8)  33335.5 (85.1)
#>                 Yes                7050.7 (18.2)   5824.1 (14.9)
#>   diab (%)      No                36326.4 (93.7)  37499.3 (95.8)
#>                 Yes                2436.8 ( 6.3)   1660.2 ( 4.2)
```

```
#>   OA (%)        Control           33060.8 (85.3)  34579.5 (88.3)
#>                 OA                 5702.3 (14.7)   4580.0 (11.7)
#>   immigrate (%) not immigrant     36812.8 (95.0)  37425.9 (95.6)
#>                 recent             1950.3 ( 5.0)   1733.7 ( 4.4)
#>                Stratified by phyact
#>                 Active          SMD
#>   n             38211.2
#>   age (%)        6567.9 (17.2)   0.145
#>                  8208.6 (21.5)
#>                  7313.3 (19.1)
#>                  5555.3 (14.5)
#>                  2186.9 ( 5.7)
#>                  4539.2 (11.9)
#>                  3840.0 (10.0)
#>   sex (%)       18441.4 (48.3)   0.077
#>                 19769.8 (51.7)
#>   married (%)   20012.7 (52.4)   0.033
#>                 18198.5 (47.6)
#>   race (%)       3397.2 ( 8.9)   0.014
#>                 34814.0 (91.1)
#>   edu (%)        6186.6 (16.2)   0.081
#>                  6159.6 (16.1)
#>                  3630.7 ( 9.5)
#>                 22234.2 (58.2)
#>   income (%)     6758.9 (17.7)   0.120
#>                  7420.8 (19.4)
#>                 10612.9 (27.8)
#>                 13418.7 (35.1)
#>   bmi (%)        1041.5 ( 2.7)   0.112
#>                 19648.4 (51.4)
#>                 17521.2 (45.9)
#>   doctor (%)     5001.8 (13.1)   0.015
#>                 33209.4 (86.9)
#>   stress (%)    30845.7 (80.7)   0.059
#>                  7365.5 (19.3)
#>   smoke (%)      8531.3 (22.3)   0.083
#>                 16200.7 (42.4)
#>                 13479.2 (35.3)
#>   drink (%)     32817.1 (85.9)   0.101
#>                  3703.9 ( 9.7)
```

```
#>                    1690.1 ( 4.4)
#>   fruit (%)        7924.2 (20.7)   0.120
#>                   19284.4 (50.5)
#>                   11002.5 (28.8)
#>   bp (%)          33698.1 (88.2)   0.120
#>                    4513.1 (11.8)
#>   diab (%)        36930.8 (96.6)   0.092
#>                    1280.4 ( 3.4)
#>   OA (%)          34640.5 (90.7)   0.110
#>                    3570.7 ( 9.3)
#>   immigrate (%) 36764.3 (96.2)   0.040
#>                    1446.8 ( 3.8)
```

All covariates are balanced in terms of SMD.

```
plot(ps.fit3, color = TRUE, plots = 2, figureRows = 1)
```

```
#> [[1]]
```

Inactive propensity scores by Tx group
Propensity scores
1.0
0.8
0.6
0.4
0.2
0.0
Inactive
Moderate
Active
Treatment

```
#>
#> [[2]]
```

Moderate propensity scores by Tx group
Propensity scores
1.0
0.8
0.6
0.4
0.2
0.0
Inactive
Moderate
Active
Treatment

```
#>
#> [[3]]
```

```
plot(ps.fit3, plots = 3, color=TRUE, pairwiseMax = FALSE)
```

```
#> [[1]]
```

Balance of Inactive versus Moderate
es.max
Absolute standard difference
0.0
0.1
0.2
0.3
0.4
Unweighted
Weighted

```
#>
#> [[2]]
```

Balance of Inactive versus Active
es.max
Absolute standard difference
0.0
0.1
0.2
0.3
0.4
Unweighted
Weighted

```
#>
#> [[3]]
```

0.0.0.0.4 * Outcome analysis

Now, we will fit the outcome model:

```
# Create an indicator variable in the full dataset
dat.full$ind <- 0
dat.full$ind[dat.full$ID %in% dat.analytic$ID] <- 1

# New weight = IPW * survey weight
dat.analytic$ATEweight3 <- with(dat.analytic, ipw3 * weight)

# New weight variable in the full dataset
dat.full$ATEweight3 <- 0
dat.full$ATEweight3[dat.full$ID %in% dat.analytic$ID] <- dat.analytic$ATEweight3

# Survey setup with full data
w.design0 <- svydesign(id = ~1, weights = ~ATEweight3, data = dat.full)

# Subset the design for analytic sample
w.design1 <- subset(w.design0, ind == 1)

# Weighted proportion
w.prop <- svyby(formula = ~CVD, by = ~phyact, design = w.design1, FUN = svymean)
w.prop
#>            phyact     CVDNo     CVDYes    se.CVDNo   se.CVDYes
```

```
#> Inactive Inactive 0.9620985 0.03790148 0.001627655 0.001627655
#> Moderate Moderate 0.9685547 0.03144532 0.002361135 0.002361135
#> Active     Active 0.9748839 0.02511610 0.002074074 0.002074074

# Outcome model
fit3 <- svyglm(CVD ~ phyact, design = w.design1, family = binomial)
publish(fit3, confint.method = "robust", pvalue.method = "robust")
#>  Variable    Units OddsRatio       CI.95   p-value
#>    phyact Inactive       Ref
#>           Moderate      0.82 [0.56;1.21]   0.32243
#>             Active      0.65 [0.44;0.96]   0.03204
```

Other approches for multiple treatments

Not covered here, but possible to do in a multiple treatments context:

- Propensity score matching
- Propensity score stratification
- Marginal mean weighting

R functions (S)

The list of new R functions introduced in this *Propensity score analyis* lab component are below:

Function_name	Package_name	Use
bal.plot	cobalt	To produce a overalp/balance plot for propensity scoes
bal.tab	cobalt	To check the balance at each category of covariates
CreateCatTable	tableone	To create a frequency table with categorical variables only
do.call	base	To execute a function call
love.plot	cobalt	To plot the standardized mean differences at each category of covariates
match.data	MatchIt	To extract the matched dataste from a matchit object
matchit	MatchIt	To match an exposed/treated to m unexposed/controls. The argument 'ratio
rownames	base	Names of the rows

Quiz (S)

> ⚠ Warning
>
> We are currently in the beta testing phase for our quiz page updates, so please be aware that some responses may not be accurate. Thank you for your patience.
> For those interested in downloading the quiz questions, you'll find a section below dedicated to downloading.

Live Quiz

You can click on the live quiz options to either see hints of why your response may be incorrect (response marked in red), or whether your choice of response is correct (response marked in green):

Download Quiz

1. **Downloading the File:**
   - Navigate to the link: See here.
   - Right-click on the link and select "Save link as…" from the dropdown menu.
   - Alternatively click here for downloading the quizzes.
   - Choose a destination folder on your computer where you'd like to save the file (e.g., Desktop). **Remember this location**, as you'll need to navigate to it later.
2. **Setting Up RStudio:**

- If you don't have RStudio installed, see the download link in here.
- Launch RStudio after installing.

3. **Installing Necessary R Packages:**

   - Before running the R Markdown file, ensure you have the required packages. In RStudio's console (located at the bottom left by default), enter the following commands to install them:

```r
install.packages("learnr")
install.packages("xfun")
```

4. **Opening and Running the File:**

   - In RStudio, go to `File` > `Open File...`.
   - Navigate to the folder where you saved the `Rmd` file and select it to open.
   - Once the file is open in RStudio, you'll see a "Run Document" button (green) at the top of the script editor. Click on it to run the R Markdown Quiz file.

App (S)

Below is an example of an app that utilizes the NHANES dataset following the tutorial materials. Users can visualize the results from a propensity score analysis using complex survey data, and implications of choosing different list of covariates for the propensity score model fit in each step of the analysis.

> Tip
>
> **Web App may need some time to load the necessary files, depending on your internet speed.**

> Important
>
> If you want to run the app locally in your computer instead for faster load time: you can download the app directly in your R via `shiny::runGitHub("EpiShinyLiveS", "ehsanx", subdir = "app")`. Make sure that `shiny`, `ggplot2`, `survey`, `tableone`, `tibble`, `dplyr`, `tidyr`, `broom.mixed` and `jtools` packages are installed in your computer/R.

> Warning
>
> **Bug Report**:
> Fill out this form to report any issues with the app.

Exercise (S)

> Tip
>
> You can download all of the related files in a zip file **propensityscoreEx.zip** from Github folder, or just by clicking this link directly.
>
> - Navigate to the GitHub folder (above link) where the ZIP file is located.
> - Click on the file name (above zip file) to open its preview window.
> - Click on the Download button to download the file. If you can't see the Download button, click on "Download Raw File" link that should appear on the page.

Problem Statement

We will use the article by Moon et al. (2021):

We will reproduce some results from the article. The authors aggregated 4 NHANES cycles 2005-12 to create their analytic dataset. The full dataset contains 40,790 subjects with the following relevant variables for this exercise:

Survey information

- SEQN: Respondent sequence number
- strata: Masked pseudo strata (strata is nested within PSU)
- psu: Masked pseudo PSU
- survey.weight: Full sample 8 year interview weight divided by 4

- survey.cycle: NHANES cycle

Outcome variable

- cvd: Cardiovascular disease

Exposure

- nocturia: Binary nocturia

Confounders and other variables

- age: Age in years at screening
- gender: Gender
- race: Race/Ethnicity
- smoking: 100+ cigarettes in life
- alcohol: Alcohol consumption (12+ drinks in 1 year)
- sleep: Sleep duration, h
- bmi: Body Mass Index in kg/m^2
- systolic: Systolic blood pressure, mmHg
- diastolic: Diastolic blood pressure, mmHg
- tcholesterol: Total cholesterol, mg/dl
- triglycerides: Triglycerides, mg/dl
- hdl: HDL-cholesterol, mg/dl
- diabetes: Diabetes mellitus
- hypertension: Hypertension

Two important **warnings** before we start:

- In this paper, there is insufficient information to create the analytic dataset. This is mainly because of not sufficiently defining the covariates and not explicitly explaining the inclusion/exclusion criteria.
- The authors did incorrect analyses. For example, they didn't consider survey features. Since we will utilize survey features in our analysis, our results will likely be different than the results shown by the authors in Table 2.

Question 1: [0% grade]

1(a) Importing dataset

```r
load(file = "Data/propensityscore/Moon2021.RData")
```

1(b) Subsetting according to eligibility

```r
# Age 20+
dat.analytic <- dat.full[complete.cases(dat.full$age),]

# Complete outcome and exposure information
dat.analytic <- dat.analytic[complete.cases(dat.analytic$cvd),]
dat.analytic <- dat.analytic[complete.cases(dat.analytic$nocturia),]

# Keep important variables only
vars <- c(
  # Survey features
  "SEQN", "strata", "psu", "survey.weight",

  # Survey cycle
  "survey.cycle",

  # Binary exposure
  "nocturia",

  # Outcome
  "cvd",

  # Covariates
  "age", "gender", "race" , "smoking", "alcohol", "sleep", "bmi", "diabetes",
  "hypertension", "tcholesterol", "triglycerides", "hdl", "systolic", "diastolic")

dat.analytic <- dat.analytic[,vars]

# Complete case
dat.analytic <- na.omit(dat.analytic) #  N = 15,404 (numbers do not match with Fig 1)
```

```r
dim(dat.analytic)
#> [1] 15404    21
```

1(c) Run the design-adjusted logistic regression

Create the first column of Table 2 of the article, i.e., explore the relationship between binary nocturia and CVD among adults aged 20 years and more. Adjust the model for age, gender, race, body mass index, smoking status, alcohol consumption, sleep duration, hypertension, diabetes mellitus, and survey cycles.

- Hint 1: the authors did not utilize the survey features (e.g., strata, psu, survey weights). But you should utilize the survey features to answer this question.
- Hint 2: Adjust the model for age, gender, race, bmi, smoking, alcohol, sleep, tcholesterol, triglycerides, hdl, hypertension, diabetes, and survey cycle.
- Hint 3: Use Publish package to report the odds ratio with the 95% CI and p-value.

```r
# Create an indicator variable in the full data
dat.full$miss <- 1
dat.full$miss[dat.full$SEQN %in% dat.analytic$SEQN] <- 0

# Design setup
svy.design0 <- svydesign(strata = ~strata, id = ~psu, weights = ~survey.weight,
                         data = dat.full, nest = TRUE)

# Subset the design
svy.design <- subset(svy.design0, miss == 0)

# Design-adjusted logistic
fit.logit <- svyglm(I(cvd == "Yes") ~ nocturia + age + gender + race + bmi +
                      smoking + alcohol + sleep + tcholesterol + triglycerides +
                      hdl + hypertension + diabetes + survey.cycle,
                    family = binomial, design = svy.design)
publish(fit.logit)
#>       Variable              Units OddsRatio         CI.95     p-value
```

```
#>       nocturia                <2       Ref
#>                               2+      1.44   [1.21;1.71]   0.0001496
#>            age            [20,40)       Ref
#>                           [40,60)      4.21   [3.05;5.82]     < 1e-04
#>                           [60,80)     11.46  [7.89;16.64]     < 1e-04
#>                          [80,Inf)     25.28 [17.51;36.50]     < 1e-04
#>         gender               Male       Ref
#>                            Female      0.68   [0.58;0.79]     < 1e-04
#>           race          Hispanics       Ref
#>                Non-Hispanic White      1.32   [1.10;1.57]   0.0036168
#>                Non-Hispanic Black      1.15   [0.92;1.44]   0.2362499
#>                       Other races      1.55   [1.05;2.30]   0.0319116
#>            bmi                        1.02   [1.01;1.03]   0.0003273
#>        smoking                 No       Ref
#>                               Yes      1.74   [1.46;2.07]     < 1e-04
#>        alcohol                 No       Ref
#>                               Yes      0.92   [0.59;1.45]   0.7273627
#>          sleep                        0.96   [0.90;1.01]   0.1146287
#>   tcholesterol                        0.99   [0.99;0.99]     < 1e-04
#>  triglycerides                        1.00   [1.00;1.00]   0.4801803
#>            hdl                        0.99   [0.98;1.00]   0.0416900
#>   hypertension                 No       Ref
#>                               Yes      2.73   [2.27;3.29]     < 1e-04
#>       diabetes                 No       Ref
#>                               Yes      1.83   [1.51;2.22]     < 1e-04
#>   survey.cycle            2005-06       Ref
#>                           2007-08      0.84   [0.65;1.07]   0.1644272
#>                           2009-10      0.91   [0.73;1.12]   0.3793696
#>                           2011-11      0.82   [0.68;0.99]   0.0398975
```

Question 2: Propensity score matching by DuGoff et al. (2014) [50% grade]

Create the second column of Table 2 (exploring the relationship between binary nocturia and CVD; the same exposure and outcome used in Question 1) using the propensity score **1:1 matching** analysis as per DuGoff et al. (2014) recommendations.

Please read the hints carefully:

- Hint 1: You should consider all four steps in the propensity score (PS) analysis:
  - Step 1: Fit the PS model by considering survey features as covariates. Other covariates for the PS model are: age, gender, race, bmi, smoking, alcohol, sleep, tcholesterol, triglycerides, hdl, hypertension, diabetes, systolic, diastolic, and survey cycle.
  - Step 2: Match an exposed subject (nocturia $\geq$ 2 times) with a control subject (nocturia <2 times) without replacement within the caliper of 0.2 times the standard deviation of the logit of PS. Set your seed to 123.
  - Step 3: Balance checking using SMD. Consider SMD <0.1 as a good covariate balancing.
  - Step 4: Fit the outcome model on the matched data. If needed, adjust for imbalanced covariates in the outcome model. Report the odds ratio with a 95% CI. For step 4, you should utilize the survey feature as the design (NOT covariates).
- Hint 2: Compare your results with the results reported by the authors. [Expected answer: 2-3 sentences]

```
# your codes here
```

Question 3: Propensity score matching by Austin et al. (2018) [50% grade]

Create the second column of Table 2 (exploring the relationship between binary nocturia and CVD; the same exposure and outcome used in Questions 1 and 2) using the propensity score **1:4 matching** analysis as per Austin et al. (2018) recommendations.

Please read the hints carefully:

- Hint 1: You should consider all four steps in the propensity score (PS) analysis:
  - Step 1: Fit the PS model by considering survey features as design, i.e., fit the design-adjusted PS model. Other covariates for the PS model are: age, gender, race, bmi, smoking, alcohol, sleep, tcholesterol, triglycerides, hdl, hypertension, diabetes, systolic, diastolic, and survey cycle.
  - Step 2: Match an exposed subject (nocturia $\geq$ 2 times) with 4 control subjects (nocturia <2 times) with replacement within the caliper of 0.2 times the standard deviation of the logit of PS. Set your seed to 123.
  - Step 3: Balance checking using SMD. Consider SMD <0.1 as a good covariate balancing. Remember, you need to multiply matching weights and survey weights to get survey-based estimates.
  - Step 4: Fit the outcome model on the matched data. If needed, adjust for imbalanced covariates in the outcome model. Report the odds ratio with a 95% CI. For step 4, you should utilize the survey feature as the design (NOT covariates).
- Hint 2: Compare the results with Question 2. What's the overall conclusion? [Expected answer: 2-3 sentences]

```
# your codes here
```

Part IX

Machine learning (ML)

Background

The chapter encompasses a series of instructional content that sequentially explores various facets of predictive modeling and machine learning, connecting them with a previous chapter. Beginning with a tutorial that revisits the application of regression for predicting continuous outcomes, it underscores the importance of understanding prediction error and overfitting, and introduces foundational machine learning concepts. The subsequent tutorial emphasizes the pivotal role of data splitting in predictive modeling, illustrating how to partition data into training and test sets and evaluate model performance across different data scenarios. Moving forward, the concept of cross-validation is explored, detailing the k-fold cross-validation method and demonstrating its implementation both manually and using the caret package. Another tutorial navigates through predicting binary outcomes using logistic regression, evaluating model performance using various metrics, and employing k-fold cross-validation.

The series then delves into supervised learning, exploring regularization techniques, decision trees, and ensemble methods, while employing various model evaluation metrics and cross-validation techniques. Lastly, unsupervised learning is introduced with a focus on the k-means clustering algorithm, discussing its implementation, determining the optimal number of clusters, and addressing associated challenges. Throughout, the tutorials provide practical examples, code snippets, and visual aids, offering a comprehensive and applied exploration of predictive modeling and machine learning concepts.

> **!** Important
>
> **Datasets**:
> All of the datasets used in this tutorial can be accessed from this GitHub repository folder

Overview of tutorials

Building upon the foundational concepts introduced in a previous chapter about prediction research, this chapter takes our understanding to the next level. We've already explored the fundamentals of propensity score methods, and now it is time to harness the power of machine learning and prediction techniques within a causal inference context, a journey that will ultimately lead us to the concept of double robust estimation methods

Revisiting: Explore Relationships for Continuous Outcomes

In this tutorial, the focus is on utilizing regression to predict continuous outcomes, specifically employing multiple linear regression to construct an initial prediction model. The tutorial revisits fundamental concepts related to prediction and introduces foundational ideas pertinent to machine learning, all while using a distinct dataset compared to previous tutorials. The process involves loading a dataset, defining variables, and fitting a model using linear regression. Subsequent sections delve into the creation of a design matrix, obtaining predictions, and measuring prediction error through various metrics like R^2 and RMSE. The tutorial also addresses the critical concept of overfitting, discussing its causes, consequences, and potential solutions, such as internal and external validation methods.

Revisiting: Data Spliting

This tutorial emphasizes the crucial concept of data splitting in the context of predictive modeling and machine learning, utilizing a different dataset than previous tutorials. The process begins with loading a dataset and then strategically splitting it into training and test subsets, ensuring a robust approach to model validation. A model is trained using the training data, and its performance is evaluated using various metrics, such as R^2 and RMSE, through a custom function that extracts these performance measures. This function facilitates the evaluation of the model's predictive accuracy and fit by applying it to different datasets (training, test, and the entire dataset), thereby enabling a comprehensive understanding of the model's performance across different data scenarios.

Revisiting: Cross-vaildation

This tutorial delves into the concept of cross-validation, a pivotal technique in predictive modeling and machine learning, using a distinct dataset for illustrative purposes. The process of

k-fold cross-validation is explored, wherein the data is partitioned into 'k' subsets, and the model is trained 'k' times, each time using a different subset as the test set and the remaining data as the training set. This method is employed to assess the model's predictive performance and to mitigate the risk of results being dependent on the initial data split. The tutorial demonstrates both manual calculations for individual folds and the utilization of the caret package to automate the cross-validation process, thereby providing a comprehensive overview of the method.

Revisiting: Explore Relationships for Binary Outcomes

The tutorial navigates through the concept of predicting binary outcomes using logistic regression, emphasizing the application of various model evaluation metrics and methodologies in a machine learning context. It begins by ensuring that the outcome variable is treated as a factor and then proceeds to model fitting, where logistic regression is applied to predict a binary outcome. The model's predictive performance is evaluated using metrics like the Area Under the Curve (AUC) and the Brier Score, which respectively assess the model's classification accuracy and the mean squared difference between predicted probabilities and the actual outcomes. Furthermore, the tutorial explores k-fold cross-validation using the caret package, providing a robust method to assess the model's predictive performance while avoiding overfitting. It also touches upon variable selection using stepwise regression with the Akaike Information Criterion (AIC) as a selection criterion.

Supervised Learning

This tutorial delves into the realm of supervised learning, exploring beyond statistical regression and introducing various machine learning methods tailored for both continuous and binary outcomes. The tutorial explores different regularization techniques, such as LASSO, Ridge, and Elastic Net, which are used to prevent overfitting by penalizing large coefficients in regression models. It also introduces decision trees (CART),

which provide a flexible, hierarchical approach to modeling data, and can automatically incorporate non-linear effects and interactions. The tutorial further explores ensemble methods, which combine predictions from multiple models to improve predictive accuracy. Two types of ensemble methods are discussed: Type I, which trains the same model on different samples of the data (e.g., bagging and boosting), and Type II, which trains different models on the same data (e.g., Super Learner). Various model evaluation metrics and cross-validation techniques are utilized throughout to assess and enhance the predictive performance of the models.

Unsupervised Learning

The tutorial introduces unsupervised learning, with a focus on clustering, a technique that categorizes data into distinct groups based on similarity without using predefined labels. The k-means clustering algorithm is highlighted, which partitions data into k groups by minimizing within-cluster variation, typically using the sum of squares of Euclidean distances. The algorithm iteratively assigns data points to clusters based on the mean of the data points in each cluster and recalculates the cluster means until the cluster assignments no longer change. Various examples illustrate how to apply k-means clustering to different datasets and variable combinations. The tutorial also discusses determining the optimal number of clusters, k, and addresses challenges such as the influence of outliers and the sensitivity to the initial assignment of cluster means.

Machine Learning for Health Survey Data using NHIS data

This tutorial provides a step-by-step guide to fitting machine learning models with health survey data, specifically using the National Health Interview Survey (NHIS) 2016 dataset to predict high impact chronic pain (HICP) among adults aged 65 years or older. The tutorial covers the use of (1) LASSO and (2) random forest models with sampling weights for population-level predictions. It also discusses the split-sample

approach for internal validation, though it acknowledges that cross-validation and bootstrapping may be better alternatives. The tutorial includes the exploration of the analytic dataset, weight normalization, split-sample creation, defining regression formulas, fitting LASSO models with optimal lambda, and evaluating model performance with metrics such as AUC, calibration slope, and Brier score. The same process is then repeated for fitting random forest models, and a variable importance plot is generated to identify influential predictors. Finally, a performance comparison table is provided.

Replicate Results from a Published Article

The tutorial guides users on implementing machine learning techniques using health survey data, specifically for predicting high impact chronic pain (HICP). It seeks to replicate results from a 2023 article, which utilized the National Health Interview Survey (NHIS) 2016 dataset. For simplicity, complete case data was used in this tutorial. In the original research, the authors developed prediction models for HICP and evaluated their performance within specific sociodemographic groups, such as gender, age groups, and race/ethnicity. They adopted LASSO and random forest models, applying 5-fold cross-validation. They also factored in survey weights in both models to achieve population-level predictions.

> Tip
>
> **Optional Content**:
> You'll find that some sections conclude with an optional video walkthrough that demonstrates the code. Keep in mind that the content might have been updated since these videos were recorded. Watching these videos is optional.

> Warning
>
> **Bug Report**:
> Fill out this form to report any issues with the tutorial.

References

Concepts (L)

Machine learning

Machine learning focuses on developing algorithms and models that enable computers to learn from and make predictions or decisions based on data without being explicitly programmed. In epidemiology, machine learning has several important uses and applications. This section is a very basic introduction to machine learning for Epidemiology.

Reading list

Key reference

- (Bi et al. 2019)

Following ate optional but useful references

- (E. Karim 2021)
- (Liu et al. 2019)
- (Kuhn et al. 2013)
- (James et al. 2013)
- (Vittinghoff et al. 2012)
- (Ewout W. Steyerberg 2019)

Video Lessons

Machine learning Reference

Machine learning Terminologies

What is included in this Video Lesson:

- Reference 0:08
- Types of Epidemiological models 0:32
- Analyzing Epidemiological Study data 2:44
- Machine learning 5:42
- Terminologies 9:15
- Classification of Machine learning 12:55
- Classification of Supervised learning 17:30
- Other classifications 18:42
- Popular algorithms 20:06
- Decision tree 20:50
- Shrinkage Methods 26:37
- Ensemble methods 37:04
- Variable Importance measure 40:20
- Epidemiologic applications 44:29
- Future Reading 53:17

The timestamps are also included in the YouTube video description.

Video Lesson Slides

Links

- Google Slides
- PDF Slides

References

Continuous outcome

> **!** Important
>
> This tutorial is very similar to one of the previous tutorials, but uses a different data (we used RHC data here). We are revisiting concepts related to prediction before introducing ideas related to machine learning.

In this chapter, we will talk about Regression that deals with prediction of continuous outcomes. We will use multiple linear regression to build the first prediction model.

Load dataset

```
ObsData <- readRDS(file = "Data/machinelearning/rhcAnalytic.RDS")
head(ObsData)
#>   Disease.category          Cancer Death Cardiovascular Congestive.HF Dementia
#> 1            Other Localized (Yes)     0              0             0        0
#> 2             MOSF            None     1              1             1        0
#> 3             MOSF Localized (Yes)     0              0             0        0
#> 4              ARF            None     1              0             0        0
#> 5             MOSF            None     1              0             0        0
#> 6            Other            None     0              0             1        0
#>   Psychiatric Pulmonary Renal Hepatic GI.Bleed Tumor Immunosupperssion
#> 1           0         1     0       0        0     1                 0
#> 2           0         0     0       0        0     0                 1
#> 3           0         0     0       0        0     1                 1
#> 4           0         0     0       0        0     0                 1
#> 5           0         0     0       0        0     0                 0
#> 6           0         1     0       0        0     0                 0
#>   Transfer.hx MI       age    sex       edu DASIndex APACHE.score
#> 1           0  0   [70,80)   Male 12.000000 23.50000           46
```

```
#> 2           1  0   [70,80) Female 12.000000 14.75195           50
#> 3           0  0 [-Inf,50) Female 14.069916 18.13672           82
#> 4           0  0   [70,80) Female  9.000000 22.92969           48
#> 5           0  0   [60,70)   Male  9.945259 21.05078           72
#> 6           0  0 [80, Inf) Female  8.000000 17.50000           38
#>   Glasgow.Coma.Score blood.pressure         WBC Heart.rate Respiratory.rate
#> 1                  0             41 22.09765620        124               10
#> 2                  0             63 28.89843750        137               38
#> 3                  0             57  0.04999542        130               40
#> 4                  0             55 23.29687500         58               26
#> 5                 41             65 29.69921880        125               27
#> 6                  0            115 18.00000000        134               36
#>   Temperature PaO2vs.FIO2  Albumin Hematocrit Bilirubin Creatinine Sodium
#> 1    38.69531     68.0000 3.500000   58.00000 1.0097656  1.1999512    145
#> 2    38.89844    218.3125 2.599609   32.50000 0.6999512  0.5999756    137
#> 3    36.39844    275.5000 3.500000   21.09766 1.0097656  2.5996094    146
#> 4    35.79688    156.6562 3.500000   26.29688 0.3999634  1.6999512    117
#> 5    34.79688    478.0000 3.500000   24.00000 1.0097656  3.5996094    126
#> 6    39.19531    184.1875 3.099609   30.50000 1.0097656  1.3999023    138
#>   Potassium PaCo2       PH   Weight DNR.status  Medical.insurance
#> 1  4.000000    40 7.359375 64.69995         No           Medicare
#> 2  3.299805    34 7.329102 45.69998         No Private & Medicare
#> 3  2.899902    16 7.359375  0.00000         No            Private
#> 4  5.799805    30 7.459961 54.59998         No Private & Medicare
#> 5  5.799805    17 7.229492 78.39996        Yes           Medicare
#> 6  5.399414    68 7.299805 54.89999         No           Medicare
#>   Respiratory.Diag Cardiovascular.Diag Neurological.Diag Gastrointestinal.Diag
#> 1              Yes                 Yes               No                    No
#> 2               No                  No               No                    No
#> 3               No                 Yes               No                    No
#> 4              Yes                  No               No                    No
#> 5               No                 Yes               No                    No
#> 6              Yes                  No               No                    No
#>   Renal.Diag Metabolic.Diag Hematologic.Diag Sepsis.Diag Trauma.Diag
#> 1         No             No               No          No          No
#> 2         No             No               No         Yes          No
#> 3         No             No               No          No          No
#> 4         No             No               No          No          No
#> 5         No             No               No          No          No
#> 6         No             No               No          No          No
```

```
#>   Orthopedic.Diag  race     income Length.of.Stay RHC.use
#> 1              No white Under $11k              9       0
#> 2              No white Under $11k             45       1
#> 3              No white   $25-$50k             60       1
#> 4              No white   $11-$25k             37       0
#> 5              No white Under $11k              2       1
#> 6              No white Under $11k              7       0
```

Prediction for length of stay

Now, we show the regression fitting when outcome is continuous (length of stay).

Variables

```
baselinevars <- names(dplyr::select(ObsData,
                         !c(Length.of.Stay,Death)))
baselinevars
#>  [1] "Disease.category"      "Cancer"                "Cardiovascular"
#>  [4] "Congestive.HF"         "Dementia"              "Psychiatric"
#>  [7] "Pulmonary"             "Renal"                 "Hepatic"
#> [10] "GI.Bleed"              "Tumor"                 "Immunosupperssion"
#> [13] "Transfer.hx"           "MI"                    "age"
#> [16] "sex"                   "edu"                   "DASIndex"
#> [19] "APACHE.score"          "Glasgow.Coma.Score"    "blood.pressure"
#> [22] "WBC"                   "Heart.rate"            "Respiratory.rate"
#> [25] "Temperature"           "PaO2vs.FIO2"           "Albumin"
#> [28] "Hematocrit"            "Bilirubin"             "Creatinine"
#> [31] "Sodium"                "Potassium"             "PaCo2"
#> [34] "PH"                    "Weight"                "DNR.status"
#> [37] "Medical.insurance"     "Respiratory.Diag"      "Cardiovascular.Diag"
#> [40] "Neurological.Diag"     "Gastrointestinal.Diag" "Renal.Diag"
#> [43] "Metabolic.Diag"        "Hematologic.Diag"      "Sepsis.Diag"
#> [46] "Trauma.Diag"           "Orthopedic.Diag"       "race"
#> [49] "income"                "RHC.use"
```

Model

```r
# adjust covariates
out.formula1 <- as.formula(paste("Length.of.Stay~ ",
                                 paste(baselinevars,
                                       collapse = "+")))
saveRDS(out.formula1, file = "Data/machinelearning/form1.RDS")
fit1 <- lm(out.formula1, data = ObsData)
require(Publish)
adj.fit1 <- publish(fit1, digits=1)$regressionTable
```

```r
out.formula1
#> Length.of.Stay ~ Disease.category + Cancer + Cardiovascular +
#>     Congestive.HF + Dementia + Psychiatric + Pulmonary + Renal +
#>     Hepatic + GI.Bleed + Tumor + Immunosupperssion + Transfer.hx +
#>     MI + age + sex + edu + DASIndex + APACHE.score + Glasgow.Coma.Score +
#>     blood.pressure + WBC + Heart.rate + Respiratory.rate + Temperature +
#>     PaO2vs.FIO2 + Albumin + Hematocrit + Bilirubin + Creatinine +
#>     Sodium + Potassium + PaCo2 + PH + Weight + DNR.status + Medical.insurance +
#>     Respiratory.Diag + Cardiovascular.Diag + Neurological.Diag +
#>     Gastrointestinal.Diag + Renal.Diag + Metabolic.Diag + Hematologic.Diag +
#>     Sepsis.Diag + Trauma.Diag + Orthopedic.Diag + race + income +
#>     RHC.use
adj.fit1
#>                 Variable                Units Coefficient          CI.95 p-value
#> 1            (Intercept)                           -76.8 [-139.4;-14.2]    <0.1
#> 2       Disease.category                  ARF        Ref
#> 3                                         CHF       -5.6    [-9.0;-2.2]    <0.1
#> 4                                       Other       -4.4    [-6.5;-2.3]    <0.1
#> 5                                        MOSF        2.9      [1.1;4.7]    <0.1
#> 6                 Cancer                 None        Ref
#> 7                             Localized (Yes)       -7.8    [-15.7;0.1]    <0.1
#> 8                                  Metastatic      -10.6   [-19.1;-2.1]    <0.1
#> 9         Cardiovascular                    0        Ref
#> 10                                          1        0.7     [-1.3;2.7]     0.5
#> 11         Congestive.HF                    0        Ref
#> 12                                          1       -1.8     [-3.9;0.4]     0.1
#> 13             Dementia                     0        Ref
#> 14                                          1       -1.3     [-3.6;1.1]     0.3
#> 15           Psychiatric                    0        Ref
```

```
#> 16                                        1       -0.5     [-3.1;2.2]     0.7
#> 17             Pulmonary                  0        Ref
#> 18                                        1        2.1      [0.1;4.0]    <0.1
#> 19                 Renal                  0        Ref
#> 20                                        1       -6.9   [-10.8;-3.1]    <0.1
#> 21               Hepatic                  0        Ref
#> 22                                        1       -1.5     [-5.1;2.1]     0.4
#> 23              GI.Bleed                  0        Ref
#> 24                                        1       -5.1    [-9.7;-0.5]    <0.1
#> 25                 Tumor                  0        Ref
#> 26                                        1        4.6    [-3.4;12.6]     0.3
#> 27     Immunosupperssion                  0        Ref
#> 28                                        1        0.1     [-1.4;1.7]     0.9
#> 29           Transfer.hx                  0        Ref
#> 30                                        1        1.2     [-0.9;3.2]     0.3
#> 31                    MI                  0        Ref
#> 32                                        1       -1.7     [-5.3;2.0]     0.4
#> 33                   age           [-Inf,50)        Ref
#> 34                                  [50,60)        0.1     [-2.0;2.3]     0.9
#> 35                                  [60,70)       -0.4     [-2.5;1.7]     0.7
#> 36                                  [70,80)       -1.1     [-3.6;1.4]     0.4
#> 37                                [80, Inf)       -2.8     [-5.7;0.2]    <0.1
#> 38                   sex               Male        Ref
#> 39                                   Female        0.8     [-0.6;2.2]     0.2
#> 40                   edu                          0.0     [-0.2;0.3]     0.7
#> 41              DASIndex                         -0.1     [-0.2;0.1]     0.4
#> 42          APACHE.score                         -0.1    [-0.1;-0.0]    <0.1
#> 43    Glasgow.Coma.Score                          0.0     [-0.0;0.0]     0.2
#> 44        blood.pressure                         -0.0     [-0.0;0.0]     0.2
#> 45                   WBC                          0.0     [-0.0;0.1]     0.2
#> 46            Heart.rate                          0.0      [0.0;0.0]    <0.1
#> 47      Respiratory.rate                         -0.0     [-0.1;0.1]     1.0
#> 48           Temperature                          0.5      [0.1;0.9]    <0.1
#> 49           PaO2vs.FIO2                         -0.0    [-0.0;-0.0]    <0.1
#> 50               Albumin                         -2.6    [-3.5;-1.6]    <0.1
#> 51            Hematocrit                         -0.2    [-0.3;-0.1]    <0.1
#> 52             Bilirubin                         -0.1     [-0.3;0.1]     0.2
#> 53            Creatinine                          0.5      [0.1;1.0]    <0.1
#> 54                Sodium                          0.1      [0.0;0.2]    <0.1
#> 55             Potassium                          0.3     [-0.3;1.0]     0.3
```

```
#> 56                PaCo2                          0.1      [0.0;0.2]    <0.1
#> 57                   PH                         10.1     [2.2;17.9]    <0.1
#> 58               Weight                          0.0     [-0.0;0.0]     1.0
#> 59           DNR.status                 No        Ref
#> 60                                     Yes      -8.0   [-10.1;-5.8]    <0.1
#> 61    Medical.insurance            Medicaid       Ref
#> 62                                 Medicare      -0.5     [-3.2;2.2]     0.7
#> 63                       Medicare & Medicaid     -2.4     [-5.8;0.9]     0.2
#> 64                              No insurance     -1.8     [-5.2;1.6]     0.3
#> 65                                   Private     -2.1     [-4.6;0.4]     0.1
#> 66                        Private & Medicare     -2.0     [-4.8;0.8]     0.2
#> 67     Respiratory.Diag                 No        Ref
#> 68                                     Yes       0.3     [-1.4;2.0]     0.7
#> 69   Cardiovascular.Diag                No        Ref
#> 70                                     Yes       0.4     [-1.4;2.1]     0.7
#> 71    Neurological.Diag                 No        Ref
#> 72                                     Yes       3.5      [1.1;6.0]    <0.1
#> 73 Gastrointestinal.Diag                No        Ref
#> 74                                     Yes       2.6      [0.3;4.8]    <0.1
#> 75           Renal.Diag                No        Ref
#> 76                                     Yes       1.8     [-1.4;5.0]     0.3
#> 77       Metabolic.Diag                No        Ref
#> 78                                     Yes      -1.2     [-4.3;2.0]     0.5
#> 79     Hematologic.Diag                No        Ref
#> 80                                     Yes      -3.9    [-6.8;-0.9]    <0.1
#> 81          Sepsis.Diag                No        Ref
#> 82                                     Yes       0.0     [-2.0;2.0]     1.0
#> 83          Trauma.Diag                No        Ref
#> 84                                     Yes       1.1     [-5.9;8.2]     0.8
#> 85      Orthopedic.Diag                No        Ref
#> 86                                     Yes       3.5   [-15.1;22.2]     0.7
#> 87                 race             white        Ref
#> 88                                  black      -1.1     [-3.1;0.8]     0.2
#> 89                                  other       0.2     [-2.5;3.0]     0.9
#> 90               income          $11-$25k        Ref
#> 91                               $25-$50k       2.5      [0.2;4.7]    <0.1
#> 92                                 > $50k       0.4     [-2.4;3.3]     0.8
#> 93                             Under $11k      -0.4     [-2.2;1.4]     0.6
#> 94               RHC.use                        2.9      [1.4;4.4]    <0.1
```

Design Matrix

- Notations
  - n is number of observations
  - p is number of covariates

Expands factors to a set of dummy variables.

```r
dim(ObsData)
#> [1] 5735   52
length(attr(terms(out.formula1), "term.labels"))
#> [1] 50
```

```r
head(model.matrix(fit1))
#>   (Intercept) Disease.categoryCHF Disease.categoryOther Disease.categoryMOSF
#> 1           1                   0                    1                    0
#> 2           1                   0                    0                    1
#> 3           1                   0                    0                    1
#> 4           1                   0                    0                    0
#> 5           1                   0                    0                    1
#> 6           1                   0                    1                    0
#>   CancerLocalized (Yes) CancerMetastatic Cardiovascular1 Congestive.HF1
#> 1                     1                0               0              0
#> 2                     0                0               1              1
#> 3                     1                0               0              0
#> 4                     0                0               0              0
#> 5                     0                0               0              0
#> 6                     0                0               0              1
#>   Dementia1 Psychiatric1 Pulmonary1 Renal1 Hepatic1 GI.Bleed1 Tumor1
#> 1         0            0          1      0        0         0      1
#> 2         0            0          0      0        0         0      0
#> 3         0            0          0      0        0         0      1
#> 4         0            0          0      0        0         0      0
#> 5         0            0          0      0        0         0      0
#> 6         0            0          1      0        0         0      0
#>   Immunosupperssion1 Transfer.hx1 MI1 age[50,60) age[60,70) age[70,80)
#> 1                  0            0   0          0          0          1
#> 2                  1            1   0          0          0          1
#> 3                  1            0   0          0          0          0
#> 4                  1            0   0          0          0          1
```

```
#> 5                  0            0   0          0          1          0
#> 6                  0            0   0          0          0          0
#>   age[80, Inf) sexFemale       edu DASIndex APACHE.score Glasgow.Coma.Score
#> 1            0         0 12.000000 23.50000           46                  0
#> 2            0         1 12.000000 14.75195           50                  0
#> 3            0         1 14.069916 18.13672           82                  0
#> 4            0         1  9.000000 22.92969           48                  0
#> 5            0         0  9.945259 21.05078           72                 41
#> 6            1         1  8.000000 17.50000           38                  0
#>   blood.pressure         WBC Heart.rate Respiratory.rate Temperature
#> 1             41 22.09765620        124               10    38.69531
#> 2             63 28.89843750        137               38    38.89844
#> 3             57  0.04999542        130               40    36.39844
#> 4             55 23.29687500         58               26    35.79688
#> 5             65 29.69921880        125               27    34.79688
#> 6            115 18.00000000        134               36    39.19531
#>   PaO2vs.FIO2  Albumin Hematocrit Bilirubin Creatinine Sodium Potassium PaCo2
#> 1     68.0000 3.500000   58.00000 1.0097656  1.1999512    145  4.000000    40
#> 2    218.3125 2.599609   32.50000 0.6999512  0.5999756    137  3.299805    34
#> 3    275.5000 3.500000   21.09766 1.0097656  2.5996094    146  2.899902    16
#> 4    156.6562 3.500000   26.29688 0.3999634  1.6999512    117  5.799805    30
#> 5    478.0000 3.500000   24.00000 1.0097656  3.5996094    126  5.799805    17
#> 6    184.1875 3.099609   30.50000 1.0097656  1.3999023    138  5.399414    68
#>         PH   Weight DNR.statusYes Medical.insuranceMedicare
#> 1 7.359375 64.69995             0                         1
#> 2 7.329102 45.69998             0                         0
#> 3 7.359375  0.00000             0                         0
#> 4 7.459961 54.59998             0                         0
#> 5 7.229492 78.39996             1                         1
#> 6 7.299805 54.89999             0                         1
#>   Medical.insuranceMedicare & Medicaid Medical.insuranceNo insurance
#> 1                                    0                            0
#> 2                                    0                            0
#> 3                                    0                            0
#> 4                                    0                            0
#> 5                                    0                            0
#> 6                                    0                            0
#>   Medical.insurancePrivate Medical.insurancePrivate & Medicare
#> 1                        0                                   0
#> 2                        0                                   1
```

```
#> 3                        1                                0
#> 4                        0                                1
#> 5                        0                                0
#> 6                        0                                0
#>   Respiratory.DiagYes Cardiovascular.DiagYes Neurological.DiagYes
#> 1                   1                      1                    0
#> 2                   0                      0                    0
#> 3                   0                      1                    0
#> 4                   1                      0                    0
#> 5                   0                      1                    0
#> 6                   1                      0                    0
#>   Gastrointestinal.DiagYes Renal.DiagYes Metabolic.DiagYes Hematologic.DiagYes
#> 1                        0             0                 0                    0
#> 2                        0             0                 0                    0
#> 3                        0             0                 0                    0
#> 4                        0             0                 0                    0
#> 5                        0             0                 0                    0
#> 6                        0             0                 0                    0
#>   Sepsis.DiagYes Trauma.DiagYes Orthopedic.DiagYes raceblack raceother
#> 1              0              0                  0         0         0
#> 2              1              0                  0         0         0
#> 3              0              0                  0         0         0
#> 4              0              0                  0         0         0
#> 5              0              0                  0         0         0
#> 6              0              0                  0         0         0
#>   income$25-$50k income> $50k incomeUnder $11k RHC.use
#> 1              0            0                1       0
#> 2              0            0                1       1
#> 3              1            0                0       1
#> 4              0            0                0       0
#> 5              0            0                1       1
#> 6              0            0                1       0
dim(model.matrix(fit1))
#> [1] 5735   64
p <- dim(model.matrix(fit1))[2] # intercept + slopes
p
#> [1] 64
```

Obtain prediction

```r
obs.y <- ObsData$Length.of.Stay
summary(obs.y)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    2.00    7.00   14.00   21.56   25.00  394.00
# Predict the above fit on ObsData data
pred.y1 <- predict(fit1, ObsData)
summary(pred.y1)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  -32.76   16.62   21.96   21.56   26.73   42.67
n <- length(pred.y1)
n
#> [1] 5735
plot(obs.y,pred.y1)
lines(lowess(obs.y,pred.y1), col = "red")
```

Measuring prediction error

Prediction error measures how well the model can predict the outcome for new data that were **not** used in developing the prediction model.

- Bias reduced for models with more variables

- Unimportant variables lead to noise / variability
- Bias variance trade-off / need penalization

R2

The provided information describes a statistical context involving a dataset of `n` values, $y_1, ..., y_n$ (referred to as y_i or as a vector $y = [y_1, ..., y_n]^T$), each paired with a fitted value $f_1, ..., f_n$ (denoted as f_i or sometimes $\hat{y}_i$, and as a vector f). The residuals, represented as e_i, are defined as the differences between the observed and the fitted values: $ e_i = y_i − f_i$

The mean of the observed data is denoted by

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

The variability of the dataset can be quantified using two sums of squares formulas: 1. **Residual Sum of Squares (SSres)** or `SSE`: It quantifies the variance remaining in the data after fitting a model, calculated as:

$$SS_{res} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

2. **Total Sum of Squares (SStot)** or `SST`: It represents the total variance in the observed data, calculated as:

$$SS_{tot} = \sum_i (y_i - \bar{y})^2$$

The **Coefficient of Determination (R^2)** or `R.2`, which provides a measure of how well the model's predictions match the observed data, is defined as:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

In the ideal scenario where the model fits the data perfectly, we have $SS_{res} = 0$ and thus $R^2 = 1$. Conversely, a baseline model, which always predicts the mean $\bar{y}$ of the observed data, would yield $R^2 = 0$. Models performing worse than this baseline

model would result in a negative R^2 value. This metric is widely utilized in regression analysis to evaluate model performance, where a higher R^2 indicates a better fit of the model to the data.

```
# Find SSE
SSE <- sum( (obs.y - pred.y1)^2 )
SSE
#> [1] 3536398
# Find SST
mean.obs.y <- mean(obs.y)
SST <- sum( (obs.y - mean.obs.y)^2 )
SST
#> [1] 3836690
# Find R2
R.2 <- 1- SSE/SST
R.2
#> [1] 0.07826832
require(caret)
caret::R2(pred.y1, obs.y)
#> [1] 0.07826832
```

ref

RMSE

```
# Find RMSE
Rmse <- sqrt(SSE/(n-p))
Rmse
#> [1] 24.97185
caret::RMSE(pred.y1, obs.y)
#> [1] 24.83212
```

See (Wikipedia 2023d)

Adj R2

The **Adjusted R^2** statistic modifies the R^2 value to counteract the automatic increase of R^2 when extra explanatory variables

are added to a model, even if they do not improve the model fit. This adjustment is crucial for ensuring that the metric offers a reliable indication of the explanatory power of the model, especially in multiple regression where several predictors are involved.

The commonly used formula is defined as:

$$\bar{R}^2 = 1 - \frac{SS_{\text{res}}/df_{\text{res}}}{SS_{\text{tot}}/df_{\text{tot}}}$$

Where:

- SS_{res} and SS_{tot} represent the residual and total sums of squares respectively.
- df_{res} and df_{tot} refer to the degrees of freedom of the residual and total sums of squares. Usually, $df_{\text{res}} = n - p$ and $df_{\text{tot}} = n - 1$, where:
  - n signifies the sample size.
  - p denotes the number of variables in the model.

This metric plays a vital role in model selection and safeguards against overfitting by penalizing the inclusion of non-informative variables

The alternate formula is:

$$\bar{R}^2 = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

This formula modifies the R^2 value, accounting for the number of predictors and offering a more parsimonious model fit measure.

```r
# Find adj R2
adjR2 <- 1-(1-R.2)*((n-1)/(n-p-1))
adjR2
#> [1] 0.06786429
```

See (Wikipedia 2023a)

Overfitting and Optimism

- Model usually performs very well in the empirical data where the model was fitted in the same data (optimistic)
- Model performs poorly in the new data (generalization is not as good)

Causes

- Model determined by data at hand without expert opinion
- Too many model parameters (age, age^2, age^3) / predictors
- Too small dataset (training) / data too noisy

Consequences

- Overestimation of effects of predictors
- Reduction in model performance in new observations

Proposed solutions

We generally use procedures such as

- Internal validation
  - sample splitting
  - cross-validation
  - bootstrap
- External validation
  - Temporal
  - Geographical
  - Different data source to calculate same variable
  - Different disease

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

References

Data spliting

> **Important**
>
> This tutorial is very similar to one of the previous tutorials, but uses a different data (we used RHC data here). We are revisiting concepts related to prediction before introducing ideas related to machine learning.

Load dataset

```r
ObsData <- readRDS(file = "Data/machinelearning/rhcAnalytic.RDS")
head(ObsData)
#>   Disease.category          Cancer Death Cardiovascular Congestive.HF Dementia
#> 1            Other Localized (Yes)     0              0             0        0
#> 2             MOSF            None     1              1             1        0
#> 3             MOSF Localized (Yes)     0              0             0        0
#> 4              ARF            None     1              0             0        0
#> 5             MOSF            None     1              0             0        0
#> 6            Other            None     0              0             1        0
#>   Psychiatric Pulmonary Renal Hepatic GI.Bleed Tumor Immunosupperssion
#> 1           0         1     0       0        0     1                 0
#> 2           0         0     0       0        0     0                 1
#> 3           0         0     0       0        0     1                 1
#> 4           0         0     0       0        0     0                 1
#> 5           0         0     0       0        0     0                 0
#> 6           0         1     0       0        0     0                 0
#>   Transfer.hx MI       age    sex       edu DASIndex APACHE.score
#> 1           0  0   [70,80)   Male 12.000000 23.50000           46
#> 2           1  0   [70,80) Female 12.000000 14.75195           50
#> 3           0  0 [-Inf,50) Female 14.069916 18.13672           82
#> 4           0  0   [70,80) Female  9.000000 22.92969           48
```

```
#> 5           0  0   [60,70)   Male  9.945259 21.05078           72
#> 6           0  0 [80, Inf) Female  8.000000 17.50000           38
#>   Glasgow.Coma.Score blood.pressure         WBC Heart.rate Respiratory.rate
#> 1                  0             41 22.09765620        124               10
#> 2                  0             63 28.89843750        137               38
#> 3                  0             57  0.04999542        130               40
#> 4                  0             55 23.29687500         58               26
#> 5                 41             65 29.69921880        125               27
#> 6                  0            115 18.00000000        134               36
#>   Temperature PaO2vs.FIO2  Albumin Hematocrit Bilirubin Creatinine Sodium
#> 1    38.69531     68.0000 3.500000   58.00000 1.0097656  1.1999512    145
#> 2    38.89844    218.3125 2.599609   32.50000 0.6999512  0.5999756    137
#> 3    36.39844    275.5000 3.500000   21.09766 1.0097656  2.5996094    146
#> 4    35.79688    156.6562 3.500000   26.29688 0.3999634  1.6999512    117
#> 5    34.79688    478.0000 3.500000   24.00000 1.0097656  3.5996094    126
#> 6    39.19531    184.1875 3.099609   30.50000 1.0097656  1.3999023    138
#>   Potassium PaCo2       PH   Weight DNR.status  Medical.insurance
#> 1  4.000000    40 7.359375 64.69995         No           Medicare
#> 2  3.299805    34 7.329102 45.69998         No Private & Medicare
#> 3  2.899902    16 7.359375  0.00000         No            Private
#> 4  5.799805    30 7.459961 54.59998         No Private & Medicare
#> 5  5.799805    17 7.229492 78.39996        Yes           Medicare
#> 6  5.399414    68 7.299805 54.89999         No           Medicare
#>   Respiratory.Diag Cardiovascular.Diag Neurological.Diag Gastrointestinal.Diag
#> 1              Yes                 Yes               No                    No
#> 2               No                  No               No                    No
#> 3               No                 Yes               No                    No
#> 4              Yes                  No               No                    No
#> 5               No                 Yes               No                    No
#> 6              Yes                  No               No                    No
#>   Renal.Diag Metabolic.Diag Hematologic.Diag Sepsis.Diag Trauma.Diag
#> 1         No             No               No          No          No
#> 2         No             No               No         Yes          No
#> 3         No             No               No          No          No
#> 4         No             No               No          No          No
#> 5         No             No               No          No          No
#> 6         No             No               No          No          No
#>   Orthopedic.Diag  race     income Length.of.Stay RHC.use
#> 1              No white Under $11k              9       0
#> 2              No white Under $11k             45       1
```

```
#> 3              No white   $25-$50k            60       1
#> 4              No white   $11-$25k            37       0
#> 5              No white Under $11k             2       1
#> 6              No white Under $11k             7       0
```

See (KDnuggets 2023; Kuhn 2023b)

```r
# Using a seed to randomize in a reproducible way
set.seed(123)
require(caret)
split<-createDataPartition(y = ObsData$Length.of.Stay,
                           p = 0.7, list = FALSE)
str(split)
#>  int [1:4017, 1] 1 2 3 4 5 6 7 8 9 10 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : NULL
#>   ..$ : chr "Resample1"
dim(split)
#> [1] 4017    1
dim(ObsData)*.7 # approximate train data
#> [1] 4014.5   36.4
dim(ObsData)*(1-.7) # approximate train data
#> [1] 1720.5   15.6
```

Split the data

```r
# create train data
train.data<-ObsData[split,]
dim(train.data)
#> [1] 4017   52
# create test data
test.data<-ObsData[-split,]
dim(test.data)
#> [1] 1718   52
```

Train the model

```
out.formula1 <- readRDS(file = "Data/machinelearning/form1.RDS")
out.formula1
#> Length.of.Stay ~ Disease.category + Cancer + Cardiovascular +
#>     Congestive.HF + Dementia + Psychiatric + Pulmonary + Renal +
#>     Hepatic + GI.Bleed + Tumor + Immunosupperssion + Transfer.hx +
#>     MI + age + sex + edu + DASIndex + APACHE.score + Glasgow.Coma.Score +
#>     blood.pressure + WBC + Heart.rate + Respiratory.rate + Temperature +
#>     PaO2vs.FIO2 + Albumin + Hematocrit + Bilirubin + Creatinine +
#>     Sodium + Potassium + PaCo2 + PH + Weight + DNR.status + Medical.insurance +
#>     Respiratory.Diag + Cardiovascular.Diag + Neurological.Diag +
#>     Gastrointestinal.Diag + Renal.Diag + Metabolic.Diag + Hematologic.Diag +
#>     Sepsis.Diag + Trauma.Diag + Orthopedic.Diag + race + income +
#>     RHC.use
fit.train1<-lm(out.formula1, data = train.data)
# summary(fit.train1)
```

Function that gives performance measures

```
perform <- function(new.data,
                    model.fit,model.formula=NULL,
                    y.name = "Y",
                    digits=3){
  # data dimension
  p <- dim(model.matrix(model.fit))[2]
  # predicted value
  pred.y <- predict(model.fit, new.data)
  # sample size
  n <- length(pred.y)
  # outcome
  new.data.y <- as.numeric(new.data[,y.name])
  # R2
  R2 <- caret:::R2(pred.y, new.data.y)
  # adj R2 using alternate formula
  df.residual <- n-p
  adjR2 <- 1-(1-R2)*((n-1)/df.residual)
  # RMSE
```

```r
  RMSE <-  caret:::RMSE(pred.y, new.data.y)
  # combine all of the results
  res <- round(cbind(n,p,R2,adjR2,RMSE),digits)
  # returning object
  return(res)
}
```

Extract performance measures

```r
perform(new.data=train.data,
        y.name = "Length.of.Stay",
        model.fit=fit.train1)
#>         n  p    R2 adjR2   RMSE
#> [1,] 4017 64 0.081 0.067 24.647
perform(new.data=test.data,
        y.name = "Length.of.Stay",
        model.fit=fit.train1)
#>         n  p    R2 adjR2   RMSE
#> [1,] 1718 64 0.056  0.02 25.488
perform(new.data=ObsData,
        y.name = "Length.of.Stay",
        model.fit=fit.train1)
#>         n  p    R2 adjR2   RMSE
#> [1,] 5735 64 0.073 0.063 24.902
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

References

Cross-validation

> **Important**
>
> This tutorial is very similar to one of the previous tutorials, but uses a different data (we used RHC data here). We are revisiting concepts related to prediction before introducing ideas related to machine learning.

Now, we will describe the ideas of cross-validation.

Load previously saved data

```
ObsData <- readRDS(file = "Data/machinelearning/rhcAnalytic.RDS")
out.formula1 <- readRDS(file = "Data/machinelearning/form1.RDS")
```

k-fold cross-vaildation

See (Wikipedia 2023b)

```r
k = 5
dim(ObsData)
#> [1] 5735   52
set.seed(567)
# create folds (based on outcome)
folds <- createFolds(ObsData$Length.of.Stay, k = k,
                     list = TRUE, returnTrain = TRUE)
mode(folds)
#> [1] "list"
dim(ObsData)*4/5 # approximate training data size
#> [1] 4588.0   41.6
dim(ObsData)/5  # approximate test data size
#> [1] 1147.0   10.4
length(folds[[1]])
#> [1] 4588
length(folds[[5]])
#> [1] 4587
str(folds[[1]])
#>  int [1:4588] 1 2 4 6 7 8 9 10 11 13 ...
str(folds[[5]])
#>  int [1:4587] 1 3 5 6 7 8 10 11 12 13 ...
```

Calculation for Fold 1

```r
fold.index <- 1
fold1.train.ids <- folds[[fold.index]]
head(fold1.train.ids)
#> [1] 1 2 4 6 7 8
fold1.train <- ObsData[fold1.train.ids,]
fold1.test <- ObsData[-fold1.train.ids,]
out.formula1
#> Length.of.Stay ~ Disease.category + Cancer + Cardiovascular +
#>     Congestive.HF + Dementia + Psychiatric + Pulmonary + Renal +
#>     Hepatic + GI.Bleed + Tumor + Immunosupperssion + Transfer.hx +
#>     MI + age + sex + edu + DASIndex + APACHE.score + Glasgow.Coma.Score +
#>     blood.pressure + WBC + Heart.rate + Respiratory.rate + Temperature +
#>     PaO2vs.FIO2 + Albumin + Hematocrit + Bilirubin + Creatinine +
#>     Sodium + Potassium + PaCo2 + PH + Weight + DNR.status + Medical.insurance +
```

```
#>     Respiratory.Diag + Cardiovascular.Diag + Neurological.Diag +
#>     Gastrointestinal.Diag + Renal.Diag + Metabolic.Diag + Hematologic.Diag +
#>     Sepsis.Diag + Trauma.Diag + Orthopedic.Diag + race + income +
#>     RHC.use
model.fit <- lm(out.formula1, data = fold1.train)
predictions <- predict(model.fit,
                       newdata = fold1.test)
perform(new.data=fold1.test,
        y.name = "Length.of.Stay",
        model.fit=model.fit)
#>         n  p    R2  adjR2  RMSE
#> [1,] 1147 64 0.051 -0.004 24.86
```

Calculation for Fold 2

```
fold.index <- 2
fold1.train.ids <- folds[[fold.index]]
head(fold1.train.ids)
#> [1] 2 3 4 5 6 7
fold1.train <- ObsData[fold1.train.ids,]
fold1.test <- ObsData[-fold1.train.ids,]
model.fit <- lm(out.formula1, data = fold1.train)
predictions <- predict(model.fit,
                       newdata = fold1.test)
perform(new.data=fold1.test,
        y.name = "Length.of.Stay",
        model.fit=model.fit)
#>         n  p    R2 adjR2   RMSE
#> [1,] 1147 64 0.066 0.011 24.714
```

Using caret package to automate

See (Kuhn 2023c)

```
# Using Caret package
set.seed(504)
# make a 5-fold CV
```

```r
ctrl<-trainControl(method = "cv",number = 5)
# fit the model with formula = out.formula1
# use training method lm
fit.cv<-train(out.formula1, trControl = ctrl,
               data = ObsData, method = "lm")
fit.cv
#> Linear Regression
#>
#> 5735 samples
#>   50 predictor
#>
#> No pre-processing
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 4588, 4588, 4587, 4589, 4588
#> Resampling results:
#>
#>   RMSE      Rsquared    MAE
#>   25.05478  0.05980578  15.19515
#>
#> Tuning parameter 'intercept' was held constant at a value of TRUE
# extract results from each test data
summary.res <- fit.cv$resample
summary.res
#>       RMSE   Rsquared      MAE Resample
#> 1 22.45199 0.06463766 14.52080    Fold1
#> 2 27.05869 0.06799916 15.29290    Fold2
#> 3 27.65794 0.06034484 15.51895    Fold3
#> 4 24.55357 0.03892546 15.47073    Fold4
#> 5 23.55174 0.06712180 15.17238    Fold5
mean(fit.cv$resample$Rsquared)
#> [1] 0.05980578
sd(fit.cv$resample$Rsquared)
#> [1] 0.01204451
mean(fit.cv$resample$RMSE)
#> [1] 25.05478
sd(fit.cv$resample$RMSE)
#> [1] 2.240366
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

References

Binary outcome

> **Important**
>
> This tutorial is very similar to one of the previous tutorials, but uses a different data (we used RHC data here). We are revisiting concepts related to prediction before introducing ideas related to machine learning.

In this chapter, we will talk about Regression that deals with prediction of binary outcomes. We will use logistic regression to build the first prediction model.

Read previously saved data

```
ObsData <- readRDS(file = "Data/machinelearning/rhcAnalytic.RDS")
```

Outcome levels (factor)

- Label
  - Possible values of outcome

```
levels(ObsData$Death)=c("No","Yes") # this is useful for caret
# ref: https://tinyurl.com/caretbin
class(ObsData$Death)
#> [1] "factor"
table(ObsData$Death)
#>
#>   No  Yes
#> 2013 3722
```

Measuring prediction error

- Brier score
  - Brier score 0 means perfect prediction, and
  - close to zero means better prediction,
  - 1 being the worst prediction.
  - Less accurate forecasts get higher score in Brier score.
- AUC
  - The area under a ROC curve is called as a c statistics.
  - c being 0.5 means random prediction and
  - 1 indicates perfect prediction

Prediction for death

In this section, we show the regression fitting when outcome is binary (death).

Variables

```
baselinevars <- names(dplyr::select(ObsData,
                          !c(Length.of.Stay,Death)))
baselinevars
#>  [1] "Disease.category"      "Cancer"                "Cardiovascular"
#>  [4] "Congestive.HF"         "Dementia"              "Psychiatric"
#>  [7] "Pulmonary"             "Renal"                 "Hepatic"
#> [10] "GI.Bleed"              "Tumor"                 "Immunosupperssion"
#> [13] "Transfer.hx"           "MI"                    "age"
#> [16] "sex"                   "edu"                   "DASIndex"
#> [19] "APACHE.score"          "Glasgow.Coma.Score"    "blood.pressure"
#> [22] "WBC"                   "Heart.rate"            "Respiratory.rate"
#> [25] "Temperature"           "PaO2vs.FIO2"           "Albumin"
#> [28] "Hematocrit"            "Bilirubin"             "Creatinine"
#> [31] "Sodium"                "Potassium"             "PaCo2"
#> [34] "PH"                    "Weight"                "DNR.status"
#> [37] "Medical.insurance"     "Respiratory.Diag"      "Cardiovascular.Diag"
```

```
#> [40] "Neurological.Diag"     "Gastrointestinal.Diag" "Renal.Diag"
#> [43] "Metabolic.Diag"        "Hematologic.Diag"      "Sepsis.Diag"
#> [46] "Trauma.Diag"           "Orthopedic.Diag"       "race"
#> [49] "income"                "RHC.use"
```

Model

```r
# adjust covariates
out.formula2 <- as.formula(paste("Death~ ", paste(baselinevars, collapse = "+")))
saveRDS(out.formula2, file = "Data/machinelearning/form2.RDS")
fit2 <- glm(out.formula2, data = ObsData,
            family = binomial(link = "logit"))
require(Publish)
adj.fit2 <- publish(fit2, digits=1)$regressionTable
```

```r
out.formula2
#> Death ~ Disease.category + Cancer + Cardiovascular + Congestive.HF +
#>     Dementia + Psychiatric + Pulmonary + Renal + Hepatic + GI.Bleed +
#>     Tumor + Immunosupperssion + Transfer.hx + MI + age + sex +
#>     edu + DASIndex + APACHE.score + Glasgow.Coma.Score + blood.pressure +
#>     WBC + Heart.rate + Respiratory.rate + Temperature + PaO2vs.FIO2 +
#>     Albumin + Hematocrit + Bilirubin + Creatinine + Sodium +
#>     Potassium + PaCo2 + PH + Weight + DNR.status + Medical.insurance +
#>     Respiratory.Diag + Cardiovascular.Diag + Neurological.Diag +
#>     Gastrointestinal.Diag + Renal.Diag + Metabolic.Diag + Hematologic.Diag +
#>     Sepsis.Diag + Trauma.Diag + Orthopedic.Diag + race + income +
#>     RHC.use
adj.fit2
#>                 Variable                Units OddsRatio      CI.95 p-value
#> 1       Disease.category                  ARF       Ref
#> 2                                         CHF       1.0  [0.8;1.4]     0.8
#> 3                                       Other       1.6  [1.3;2.0]    <0.1
#> 4                                        MOSF       1.0  [0.9;1.2]     0.7
#> 5                 Cancer                 None       Ref
#> 6                           Localized (Yes)       6.6 [2.2;19.4]    <0.1
#> 7                                Metastatic      26.4 [8.3;83.6]    <0.1
#> 8         Cardiovascular                    0       Ref
#> 9                                           1       1.3  [1.0;1.5]    <0.1
```

```
#> 10         Congestive.HF                    0       Ref
#> 11                                          1       1.6  [1.3;1.9]    <0.1
#> 12              Dementia                    0       Ref
#> 13                                          1       1.3  [1.0;1.6]    <0.1
#> 14           Psychiatric                    0       Ref
#> 15                                          1       0.9  [0.7;1.2]     0.6
#> 16             Pulmonary                    0       Ref
#> 17                                          1       1.0  [0.9;1.2]     0.7
#> 18                 Renal                    0       Ref
#> 19                                          1       1.3  [0.9;1.9]     0.2
#> 20               Hepatic                    0       Ref
#> 21                                          1       1.3  [0.9;1.8]     0.1
#> 22              GI.Bleed                    0       Ref
#> 23                                          1       1.2  [0.8;1.9]     0.3
#> 24                 Tumor                    0       Ref
#> 25                                          1       0.3  [0.1;0.9]    <0.1
#> 26     Immunosupperssion                    0       Ref
#> 27                                          1       1.2  [1.1;1.4]    <0.1
#> 28           Transfer.hx                    0       Ref
#> 29                                          1       0.8  [0.7;1.0]    <0.1
#> 30                    MI                    0       Ref
#> 31                                          1       0.8  [0.6;1.2]     0.3
#> 32                   age            [-Inf,50)       Ref
#> 33                                 [50,60)       1.4  [1.2;1.8]    <0.1
#> 34                                 [60,70)       2.1  [1.7;2.5]    <0.1
#> 35                                 [70,80)       2.1  [1.6;2.6]    <0.1
#> 36                                [80, Inf)       2.8  [2.1;3.8]    <0.1
#> 37                   sex                Male       Ref
#> 38                                    Female       0.8  [0.7;0.9]    <0.1
#> 39                   edu                         1.0  [1.0;1.0]     0.4
#> 40              DASIndex                         1.0  [0.9;1.0]    <0.1
#> 41          APACHE.score                         1.0  [1.0;1.0]    <0.1
#> 42    Glasgow.Coma.Score                         1.0  [1.0;1.0]    <0.1
#> 43        blood.pressure                         1.0  [1.0;1.0]     0.4
#> 44                   WBC                         1.0  [1.0;1.0]     0.2
#> 45            Heart.rate                         1.0  [1.0;1.0]     0.7
#> 46      Respiratory.rate                         1.0  [1.0;1.0]     0.8
#> 47           Temperature                         0.9  [0.9;1.0]    <0.1
#> 48           PaO2vs.FIO2                         1.0  [1.0;1.0]     0.3
#> 49               Albumin                         1.0  [0.9;1.1]     0.6
```

```
#> 50            Hematocrit                          1.0  [1.0;1.0]    <0.1
#> 51             Bilirubin                          1.0  [1.0;1.1]    <0.1
#> 52            Creatinine                          1.0  [1.0;1.0]     0.9
#> 53                Sodium                          1.0  [1.0;1.0]     0.7
#> 54             Potassium                          1.0  [0.9;1.1]     0.9
#> 55                 PaCo2                          1.0  [1.0;1.0]     0.3
#> 56                    PH                          1.1  [0.5;2.3]     0.9
#> 57                Weight                          1.0  [1.0;1.0]    <0.1
#> 58            DNR.status                   No     Ref
#> 59                                        Yes     2.6  [2.0;3.3]    <0.1
#> 60     Medical.insurance            Medicaid     Ref
#> 61                                  Medicare     1.6  [1.2;2.0]    <0.1
#> 62                       Medicare & Medicaid     1.4  [1.0;1.9]    <0.1
#> 63                              No insurance     1.5  [1.1;2.0]    <0.1
#> 64                                   Private     1.3  [1.1;1.7]    <0.1
#> 65                        Private & Medicare     1.3  [1.0;1.7]    <0.1
#> 66      Respiratory.Diag                   No     Ref
#> 67                                        Yes     1.2  [1.0;1.4]    <0.1
#> 68   Cardiovascular.Diag                   No     Ref
#> 69                                        Yes     1.2  [1.0;1.4]    <0.1
#> 70     Neurological.Diag                   No     Ref
#> 71                                        Yes     1.5  [1.2;1.9]    <0.1
#> 72 Gastrointestinal.Diag                   No     Ref
#> 73                                        Yes     1.3  [1.1;1.6]    <0.1
#> 74            Renal.Diag                   No     Ref
#> 75                                        Yes     0.8  [0.6;1.1]     0.2
#> 76        Metabolic.Diag                   No     Ref
#> 77                                        Yes     1.0  [0.8;1.4]     0.8
#> 78      Hematologic.Diag                   No     Ref
#> 79                                        Yes     2.7  [2.0;3.8]    <0.1
#> 80           Sepsis.Diag                   No     Ref
#> 81                                        Yes     1.1  [0.9;1.4]     0.2
#> 82           Trauma.Diag                   No     Ref
#> 83                                        Yes     0.8  [0.4;1.4]     0.4
#> 84       Orthopedic.Diag                   No     Ref
#> 85                                        Yes     1.4  [0.2;8.1]     0.7
#> 86                  race               white     Ref
#> 87                                     black     1.0  [0.8;1.2]     0.9
#> 88                                     other     1.1  [0.8;1.4]     0.7
#> 89                income            $11-$25k     Ref
```

```
#> 90                                  $25-$50k       0.8  [0.7;1.0]    <0.1
#> 91                                    > $50k       0.8  [0.6;1.1]     0.2
#> 92                                Under $11k       1.2  [1.0;1.4]    <0.1
#> 93                RHC.use                          1.4  [1.2;1.6]    <0.1
```

Measuring prediction error

0.0.0.0.1 * AUC

```r
require(pROC)
#> Loading required package: pROC
#> Type 'citation("pROC")' for a citation.
#>
#> Attaching package: 'pROC'
#> The following objects are masked from 'package:stats':
#>
#>     cov, smooth, var
obs.y2<-ObsData$Death
pred.y2 <- predict(fit2, type = "response")
rocobj <- roc(obs.y2, pred.y2)
#> Setting levels: control = No, case = Yes
#> Setting direction: controls < cases
rocobj
#>
#> Call:
#> roc.default(response = obs.y2, predictor = pred.y2)
#>
#> Data: pred.y2 in 2013 controls (obs.y2 No) < 3722 cases (obs.y2 Yes).
#> Area under the curve: 0.7682
plot(rocobj)
```

```
auc(rocobj)
#> Area under the curve: 0.7682
```

0.0.0.0.2 * Brier Score

```
require(DescTools)
#> Loading required package: DescTools
BrierScore(fit2)
#> [1] 0.1812502
```

Cross-validation using caret

Basic setup

```
# Using Caret package
set.seed(504)

# make a 5-fold CV
require(caret)
#> Loading required package: caret
#> Loading required package: ggplot2
#> Loading required package: lattice
```

```
#>
#> Attaching package: 'caret'
#> The following objects are masked from 'package:DescTools':
#>
#>     MAE, RMSE
ctrl<-trainControl(method = "cv", number = 5,
                   classProbs = TRUE,
                   summaryFunction = twoClassSummary)

# fit the model with formula = out.formula2
# use training method glm (have to specify family)
fit.cv.bin<-train(out.formula2, trControl = ctrl,
               data = ObsData, method = "glm",
              family = binomial(),
              metric="ROC")
fit.cv.bin
#> Generalized Linear Model
#>
#> 5735 samples
#>   50 predictor
#>    2 classes: 'No', 'Yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 4587, 4589, 4587, 4589, 4588
#> Resampling results:
#>
#>   ROC        Sens       Spec
#>   0.7545115  0.4659618  0.8535653
```

Extract results from each test data

```
summary.res <- fit.cv.bin$resample
summary.res
#>         ROC      Sens      Spec Resample
#> 1 0.7444835 0.4739454 0.8630872    Fold1
#> 2 0.7544836 0.4502488 0.8561828    Fold2
#> 3 0.7786734 0.4739454 0.8738255    Fold3
```

```
#> 4 0.7350679 0.4626866 0.8373656    Fold4
#> 5 0.7598488 0.4689826 0.8373656    Fold5
mean(fit.cv.bin$resample$ROC)
#> [1] 0.7545115
sd(fit.cv.bin$resample$ROC)
#> [1] 0.01651437
```

More options

```
ctrl<-trainControl(method = "cv", number = 5,
                   classProbs = TRUE,
                   summaryFunction = twoClassSummary)
fit.cv.bin<-train(out.formula2, trControl = ctrl,
               data = ObsData, method = "glm",
              family = binomial(),
              metric="ROC",
              preProc = c("center", "scale"))
fit.cv.bin
#> Generalized Linear Model
#>
#> 5735 samples
#>   50 predictor
#>    2 classes: 'No', 'Yes'
#>
#> Pre-processing: centered (63), scaled (63)
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 4588, 4589, 4587, 4588, 4588
#> Resampling results:
#>
#>   ROC        Sens       Spec
#>   0.7548047  0.4629717  0.8530367
```

Variable selection

We can also use stepwise regression that uses AIC as a criterion.

```r
set.seed(504)
ctrl<-trainControl(method = "cv", number = 5,
                   classProbs = TRUE,
                   summaryFunction = twoClassSummary)
fit.cv.bin.aic<-train(out.formula2, trControl = ctrl,
               data = ObsData, method = "glmStepAIC",
               direction ="backward",
              family = binomial(),
              metric="ROC")
```

```r
fit.cv.bin.aic
#> Generalized Linear Model with Stepwise Feature Selection
#>
#> 5735 samples
#>   50 predictor
#>    2 classes: 'No', 'Yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 4587, 4589, 4587, 4589, 4588
#> Resampling results:
#>
#>   ROC        Sens      Spec
#>   0.7540424  0.464468  0.8562535
summary(fit.cv.bin.aic)
#>
#> Call:
#> NULL
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -2.8626  -0.9960   0.5052   0.8638   1.9578
#>
#> Coefficients:
#>                                        Estimate Std. Error z value Pr(>|z|)
#> (Intercept)                           1.0783624  0.7822168   1.379 0.168019
#> Disease.categoryOther                 0.4495099  0.0919860   4.887 1.03e-06
#> `CancerLocalized (Yes)`               1.8942512  0.5501880   3.443 0.000575
#> CancerMetastatic                      3.2703316  0.5858715   5.582 2.38e-08
#> Cardiovascular1                       0.2386749  0.0939617   2.540 0.011081
```

```
#> Congestive.HF1                          0.4539010  0.0971624   4.672 2.99e-06
#> Dementia1                               0.2380213  0.1162903   2.047 0.040679
#> Hepatic1                                0.3593093  0.1541762   2.331 0.019779
#> Tumor1                                 -1.2455123  0.5542624  -2.247 0.024630
#> Immunosupperssion1                      0.2174294  0.0730803   2.975 0.002928
#> Transfer.hx1                           -0.1849029  0.0945679  -1.955 0.050555
#> `age[50,60)`                            0.3621248  0.0984288   3.679 0.000234
#> `age[60,70)`                            0.6941924  0.0968434   7.168 7.60e-13
#> `age[70,80)`                            0.6804939  0.1126637   6.040 1.54e-09
#> `age[80, Inf)`                          0.9833851  0.1410563   6.972 3.13e-12
#> sexFemale                              -0.2805950  0.0653527  -4.294 1.76e-05
#> DASIndex                               -0.0429272  0.0062191  -6.902 5.11e-12
#> APACHE.score                            0.0174907  0.0020017   8.738  < 2e-16
#> Glasgow.Coma.Score                      0.0093657  0.0012563   7.455 9.00e-14
#> WBC                                     0.0044518  0.0030090   1.479 0.139009
#> Temperature                            -0.0524703  0.0192757  -2.722 0.006487
#> PaO2vs.FIO2                             0.0004741  0.0003054   1.552 0.120548
#> Hematocrit                             -0.0154796  0.0041593  -3.722 0.000198
#> Bilirubin                               0.0313087  0.0094004   3.331 0.000867
#> Weight                                 -0.0031548  0.0011213  -2.813 0.004902
#> DNR.statusYes                           0.9347360  0.1326924   7.044 1.86e-12
#> Medical.insuranceMedicare               0.4764895  0.1257582   3.789 0.000151
#> `Medical.insuranceMedicare & Medicaid`  0.3364916  0.1584757   2.123 0.033729
#> `Medical.insuranceNo insurance`         0.3711345  0.1568820   2.366 0.017996
#> Medical.insurancePrivate                0.2632637  0.1139805   2.310 0.020903
#> `Medical.insurancePrivate & Medicare`   0.2819715  0.1313101   2.147 0.031764
#> Respiratory.DiagYes                     0.1393974  0.0769026   1.813 0.069886
#> Cardiovascular.DiagYes                  0.1804967  0.0836679   2.157 0.030982
#> Neurological.DiagYes                    0.4320266  0.1189357   3.632 0.000281
#> Gastrointestinal.DiagYes                0.2819563  0.1092206   2.582 0.009836
#> Hematologic.DiagYes                     0.9734424  0.1651363   5.895 3.75e-09
#> Sepsis.DiagYes                          0.1539651  0.0943235   1.632 0.102614
#> `incomeUnder $11k`                      0.2151437  0.0689392   3.121 0.001804
#> RHC.use                                 0.3552053  0.0713632   4.977 6.44e-07
#>
#> (Intercept)
#> Disease.categoryOther                  ***
#> `CancerLocalized (Yes)`                ***
#> CancerMetastatic                       ***
#> Cardiovascular1                        *
```

```
#> Congestive.HF1                          ***
#> Dementia1                               *
#> Hepatic1                                *
#> Tumor1                                  *
#> Immunosupperssion1                      **
#> Transfer.hx1                            .
#> `age[50,60)`                            ***
#> `age[60,70)`                            ***
#> `age[70,80)`                            ***
#> `age[80, Inf)`                          ***
#> sexFemale                               ***
#> DASIndex                                ***
#> APACHE.score                            ***
#> Glasgow.Coma.Score                      ***
#> WBC
#> Temperature                             **
#> PaO2vs.FIO2
#> Hematocrit                              ***
#> Bilirubin                               ***
#> Weight                                  **
#> DNR.statusYes                           ***
#> Medical.insuranceMedicare               ***
#> `Medical.insuranceMedicare & Medicaid` *
#> `Medical.insuranceNo insurance`         *
#> Medical.insurancePrivate                *
#> `Medical.insurancePrivate & Medicare`  *
#> Respiratory.DiagYes                     .
#> Cardiovascular.DiagYes                  *
#> Neurological.DiagYes                    ***
#> Gastrointestinal.DiagYes                **
#> Hematologic.DiagYes                     ***
#> Sepsis.DiagYes
#> `incomeUnder $11k`                      **
#> RHC.use                                 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 7433.3  on 5734  degrees of freedom
```

```
#> Residual deviance: 6198.0  on 5696  degrees of freedom
#> AIC: 6276
#>
#> Number of Fisher Scoring iterations: 5
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Supervised learning

In this chapter, we will move beyond statistical regression, and introduce some of the popular machine learning methods.

In the first code chunk, we load necessary R libraries that will be utilized throughout the chapter for various machine learning methods and data visualization.

Read previously saved data

The second chunk is dedicated to reading previously saved data and formulas from specified file paths, ensuring that the dataset and predefined formulas are available for subsequent analyses.

```
ObsData <- readRDS(file = "Data/machinelearning/rhcAnalytic.RDS")
levels(ObsData$Death)=c("No","Yes")
out.formula1 <- readRDS(file = "Data/machinelearning/form1.RDS")
out.formula2 <- readRDS(file = "Data/machinelearning/form2.RDS")
```

Ridge, LASSO, and Elastic net

The traditional regression models (e.g., linear regression, logistic regression) show poor model performance when

- predictors are highly correlated or
- there are many predictors

In both cases, the variance of the estimated regression coefficients could be highly variable. Hence, the model often results in poor predictions. The general solution to this problem is to reduce the variance at the cost of introducing some bias in the

coefficients. This approach is called regularization or shrinking. Since we are interested in overall prediction rather than individual regression coefficients in a prediction context, this shrinkage approach is almost always beneficial for the model's predictive performance. Ridge, LASSO, and Elastic Net are shrinkage machine learning techniques.

- Ridge: Penalizes the sum of squared regression coefficients (the so-called L_2 penalty). This approach does not remove irrelevant predictors, but minimizes the impact of the irrelevant predictors. There is a hyperparameter called λ (lambda) that determines the amount of shrinkage of the coefficients. The larger λ indicates more penalization of the coefficients.
- LASSO: The Least Absolute Shrinkage and Selection Operator (LASSO) is quite similar conceptually to the ridge regression. However, lasso penalizes the sum of the absolute values of regression coefficients (so-called L_1 penalty). As a result, a high λ value forces many coefficients to be exactly zero in lasso regression, suggesting a reduced model with fewer predictors, which is never the case in ridge regression.
- Elastic Net: The elastic net combines the penalties of ridge regression and lasso to get the best of both methods. Two hyperparameters in the elastic net are α (alpha) and λ.

We can use the `glmnet` function in R to fit these there models.

> **Note**
>
> In glmnet function, `alpha = 1` for the LASSO, `alpha = 0` for the ridge, and setting alpha to some value between 0 and 1 is the elastic net model.

Continuous outcome

Cross-validation LASSO

- Regression (no shrinkage)
  - Minimize Residual SS

$$\text{RSS} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2$$

- Ridge
  - All coef = non-zero

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = \text{RSS} + \lambda\sum_{j=1}^{p}\beta_j^2$$

- LASSO
  - Assigns some coef = 0
  - Can be unstable
  - Addresses collinearity

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \text{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|$$

- Elasticnet
  - Compromise

How to estimate lambda? Cross validation.

In this code chunk, we implement a machine learning model training process with a focus on utilizing cross-validation and tuning parameters to optimize the model. Cross-validation is a technique used to assess how well the model will generalize to an independent dataset by partitioning the original dataset into a training set to train the model, and a test set to evaluate it. Here, we specify that we are using a particular type of cross-validation, denoted as "cv", and that we will be creating 5 folds (or partitions) of the data, as indicated by `number = 5`.

The model being trained is specified to use a method known as "glmnet", which is capable of performing lasso, ridge, and elastic net regularization regressions. Tuning parameters are crucial in controlling the behavior of our learning algorithm. In this instance, we specify `lambda` and `alpha` as our tuning parameters, which control the amount of regularization applied to the model and the mixing percentage between lasso and ridge regression, respectively. The `tuneGrid` argument is used to specify the exact values of `alpha` and `lambda` that the model should consider during training. The `verbose = FALSE` argument ensures that additional model training details are not printed during the training process. Finally, the trained model is stored in an object for further examination and use.

```r
ctrl <- trainControl(method = "cv", number = 5)
fit.cv.con <- train(out.formula1,
                    trControl = ctrl,
                    data = ObsData, method = "glmnet",
                    lambda= 0,
                    tuneGrid = expand.grid(alpha = 1, lambda = 0),
                    verbose = FALSE)
fit.cv.con
#> glmnet
#>
#> 5735 samples
#>   50 predictor
#>
#> No pre-processing
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 4587, 4588, 4588, 4587, 4590
#> Resampling results:
#>
#>   RMSE      Rsquared    MAE
#>   25.05241  0.06168372  15.1717
#>
#> Tuning parameter 'alpha' was held constant at a value of 1
#> Tuning
#>  parameter 'lambda' was held constant at a value of 0
```

Cross-validation Ridge

Subsequent code chunks explore Ridge regression and Elastic Net, employing similar methodologies but adjusting tuning parameters accordingly.

```r
ctrl <- trainControl(method = "cv", number = 5)
fit.cv.con <-train(out.formula1,
                   trControl = ctrl,
                   data = ObsData, method = "glmnet",
                   lambda= 0,
                   tuneGrid = expand.grid(alpha = 0, lambda = 0),
                   verbose = FALSE)
fit.cv.con
```

```
#> glmnet
#>
#> 5735 samples
#>   50 predictor
#>
#> No pre-processing
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 4588, 4588, 4588, 4588, 4588
#> Resampling results:
#>
#>   RMSE      Rsquared    MAE
#>   25.03454  0.05945909  15.17895
#>
#> Tuning parameter 'alpha' was held constant at a value of 0
#> Tuning
#>  parameter 'lambda' was held constant at a value of 0
```

Binary outcome

Cross-validation LASSO

We then shift to binary outcomes, exploring LASSO and Ridge regression with similar implementations but adjusting for the binary nature of the outcome variable.

```
ctrl<-trainControl(method = "cv", number = 5,
                   classProbs = TRUE,
                   summaryFunction = twoClassSummary)
fit.cv.bin<-train(out.formula2,
                  trControl = ctrl,
                  data = ObsData,
                  method = "glmnet",
                  lambda= 0,
                  tuneGrid = expand.grid(alpha = 1, lambda = 0),
                  verbose = FALSE,
                  metric="ROC")
fit.cv.bin
#> glmnet
#>
```

```
#> 5735 samples
#>   50 predictor
#>    2 classes: 'No', 'Yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 4588, 4588, 4588, 4588, 4588
#> Resampling results:
#>
#>   ROC        Sens       Spec
#>   0.7558135  0.4629989  0.8533055
#>
#> Tuning parameter 'alpha' was held constant at a value of 1
#> Tuning
#>  parameter 'lambda' was held constant at a value of 0
```

- Not okay to select variables from a shrinkage model, and then use them in a regular regression

Cross-validation Ridge

```
ctrl<-trainControl(method = "cv", number = 5,
                   classProbs = TRUE,
                   summaryFunction = twoClassSummary)
fit.cv.bin<-train(out.formula2, trControl = ctrl,
               data = ObsData, method = "glmnet",
               lambda= 0,
               tuneGrid = expand.grid(alpha = 0,
                                      lambda = 0),
               verbose = FALSE,
               metric="ROC")
fit.cv.bin
#> glmnet
#>
#> 5735 samples
#>   50 predictor
#>    2 classes: 'No', 'Yes'
#>
```

```
#> No pre-processing
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 4587, 4588, 4589, 4589, 4587
#> Resampling results:
#>
#>   ROC        Sens       Spec
#>   0.7537253  0.4664741  0.8538381
#>
#> Tuning parameter 'alpha' was held constant at a value of 0
#> Tuning
#>  parameter 'lambda' was held constant at a value of 0
```

Cross-validation Elastic net

- Alpha = mixing parameter
- Lambda = regularization or tuning parameter
- We can use `expand.grid` for model tuning

```
ctrl<-trainControl(method = "cv", number = 5,
                   classProbs = TRUE,
                   summaryFunction = twoClassSummary)
fit.cv.bin<-train(out.formula2, trControl = ctrl,
               data = ObsData, method = "glmnet",
               tuneGrid = expand.grid(alpha = seq(0.1,.2,by = 0.05),
                                      lambda = seq(0.05,0.3,by = 0.05)),
               verbose = FALSE,
               metric="ROC")
fit.cv.bin
#> glmnet
#>
#> 5735 samples
#>   50 predictor
#>    2 classes: 'No', 'Yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 4588, 4588, 4587, 4588, 4589
#> Resampling results across tuning parameters:
#>
```

```
#>   alpha  lambda  ROC        Sens          Spec
#>   0.10   0.05    0.7524214  0.3705739294  0.8965635
#>   0.10   0.10    0.7488141  0.2757033690  0.9357873
#>   0.10   0.15    0.7439984  0.1698986457  0.9669524
#>   0.10   0.20    0.7389996  0.0889238670  0.9881782
#>   0.10   0.25    0.7323365  0.0243435428  0.9965068
#>   0.10   0.30    0.7252894  0.0009937904  0.9997312
#>   0.15   0.05    0.7513613  0.3497105045  0.9070394
#>   0.15   0.10    0.7446604  0.2195807563  0.9497586
#>   0.15   0.15    0.7356443  0.0988642396  0.9849542
#>   0.15   0.20    0.7243101  0.0188758441  0.9970448
#>   0.15   0.25    0.7169066  0.0000000000  0.9997312
#>   0.15   0.30    0.7110961  0.0000000000  1.0000000
#>   0.20   0.05    0.7496455  0.3283508018  0.9151003
#>   0.20   0.10    0.7391386  0.1659370641  0.9680284
#>   0.20   0.15    0.7236848  0.0387467131  0.9919391
#>   0.20   0.20    0.7142421  0.0004962779  0.9997312
#>   0.20   0.25    0.7075431  0.0000000000  1.0000000
#>   0.20   0.30    0.6970569  0.0000000000  1.0000000
#>
#> ROC was used to select the optimal model using the largest value.
#> The final values used for the model were alpha = 0.1 and lambda = 0.05.
plot(fit.cv.bin)
```

Decision tree

Decision trees are then introduced and implemented, with visualizations and evaluation metrics provided to assess their performance.

- Decision tree
  - Referred to as Classification and regression trees or CART
  - Covers
    - Classification (categorical outcome)
    - Regression (continuous outcome)
  - Flexible to incorporate non-linear effects automatically
    - No need to specify higher order terms / interactions
  - Unstable, prone to overfitting, suffers from high variance

0.0.0.0.1 * Simple CART

```
require(rpart)
summary(ObsData$DASIndex) # Duke Activity Status Index
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   11.00   16.06   19.75   20.50   23.43   33.00
cart.fit <- rpart(Death~DASIndex, data = ObsData)
par(mfrow = c(1,1), xpd = NA)
plot(cart.fit)
text(cart.fit, use.n = TRUE)
```

```r
print(cart.fit)
#> n= 5735
#>
#> node), split, n, loss, yval, (yprob)
#>       * denotes terminal node
#>
#> 1) root 5735 2013 Yes (0.3510026 0.6489974)
#>   2) DASIndex>=24.92383 1143  514 No (0.5503062 0.4496938)
#>     4) DASIndex>=29.14648 561  199 No (0.6452763 0.3547237) *
#>     5) DASIndex< 29.14648 582  267 Yes (0.4587629 0.5412371) *
#>   3) DASIndex< 24.92383 4592 1384 Yes (0.3013937 0.6986063) *
require(rattle)
require(rpart.plot)
require(RColorBrewer)
fancyRpartPlot(cart.fit, caption = NULL)
```

0.0.0.0.1.1 * AUC

```
require(pROC)
obs.y2<-ObsData$Death
pred.y2 <- as.numeric(predict(cart.fit, type = "prob")[, 2])
rocobj <- roc(obs.y2, pred.y2)
#> Setting levels: control = No, case = Yes
#> Setting direction: controls < cases
rocobj
#>
#> Call:
#> roc.default(response = obs.y2, predictor = pred.y2)
#>
#> Data: pred.y2 in 2013 controls (obs.y2 No) < 3722 cases (obs.y2 Yes).
#> Area under the curve: 0.5912
plot(rocobj)
```

```
auc(rocobj)
#> Area under the curve: 0.5912
```

0.0.0.0.2 * Complex CART

More variables

```
out.formula2
#> Death ~ Disease.category + Cancer + Cardiovascular + Congestive.HF +
#>     Dementia + Psychiatric + Pulmonary + Renal + Hepatic + GI.Bleed +
#>     Tumor + Immunosupperssion + Transfer.hx + MI + age + sex +
#>     edu + DASIndex + APACHE.score + Glasgow.Coma.Score + blood.pressure +
#>     WBC + Heart.rate + Respiratory.rate + Temperature + PaO2vs.FIO2 +
#>     Albumin + Hematocrit + Bilirubin + Creatinine + Sodium +
#>     Potassium + PaCo2 + PH + Weight + DNR.status + Medical.insurance +
#>     Respiratory.Diag + Cardiovascular.Diag + Neurological.Diag +
#>     Gastrointestinal.Diag + Renal.Diag + Metabolic.Diag + Hematologic.Diag +
#>     Sepsis.Diag + Trauma.Diag + Orthopedic.Diag + race + income +
#>     RHC.use
require(rpart)
cart.fit <- rpart(out.formula2, data = ObsData)
```

0.0.0.0.3 * CART Variable importance

```r
cart.fit$variable.importance
#>            DASIndex              Cancer               Tumor                 age
#>         123.2102455          33.4559400          32.5418433          24.0804860
#>   Medical.insurance                 WBC                 edu Cardiovascular.Diag
#>          14.5199953           5.6673997           3.7441554           3.6449371
#>          Heart.rate      Cardiovascular         Trauma.Diag               PaCo2
#>           3.4059248           3.1669125           0.5953098           0.2420672
#>           Potassium              Sodium             Albumin
#>           0.2420672           0.2420672           0.1984366
```

0.0.0.0.3.1 * AUC

```r
require(pROC)
obs.y2<-ObsData$Death
pred.y2 <- as.numeric(predict(cart.fit, type = "prob")[, 2])
rocobj <- roc(obs.y2, pred.y2)
#> Setting levels: control = No, case = Yes
#> Setting direction: controls < cases
rocobj
#>
#> Call:
#> roc.default(response = obs.y2, predictor = pred.y2)
#>
#> Data: pred.y2 in 2013 controls (obs.y2 No) < 3722 cases (obs.y2 Yes).
#> Area under the curve: 0.5981
plot(rocobj)
```

```
auc(rocobj)
#> Area under the curve: 0.5981
```

0.0.0.0.4 * Cross-validation CART

```
set.seed(504)
require(caret)
ctrl<-trainControl(method = "cv", number = 5,
                   classProbs = TRUE,
                   summaryFunction = twoClassSummary)
# fit the model with formula = out.formula2
fit.cv.bin<-train(out.formula2, trControl = ctrl,
               data = ObsData, method = "rpart",
              metric="ROC")
fit.cv.bin
#> CART
#>
#> 5735 samples
#>   50 predictor
#>    2 classes: 'No', 'Yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 4587, 4589, 4587, 4589, 4588
```

```
#> Resampling results across tuning parameters:
#>
#>   cp           ROC        Sens       Spec
#>   0.007203179  0.6304911  0.2816488  0.9086574
#>   0.039741679  0.5725283  0.2488649  0.8981807
#>   0.057128664  0.5380544  0.1287804  0.9473284
#>
#> ROC was used to select the optimal model using the largest value.
#> The final value used for the model was cp = 0.007203179.
# extract results from each test data
summary.res <- fit.cv.bin$resample
summary.res
#>         ROC      Sens      Spec Resample
#> 1 0.6847220 0.3746898 0.8590604    Fold1
#> 2 0.6729625 0.2985075 0.8924731    Fold2
#> 3 0.6076153 0.2754342 0.9287634    Fold5
#> 4 0.5873154 0.2238806 0.9274194    Fold4
#> 5 0.5998401 0.2357320 0.9355705    Fold3
```

Ensemble methods (Type I)

We explore ensemble methods, specifically bagging and boosting, through implementation and evaluation in the context of binary outcomes.

Training same model to different samples (of the same data)

Cross-validation bagging

- Bagging or bootstrap aggregation
  - independent bootstrap samples (sampling with replacement, B times),
  - applies CART on each i (no prunning)
  - Average the resulting predictions
  - Reduces variance as a result of using bootstrap

```r
set.seed(504)
require(caret)
ctrl<-trainControl(method = "cv", number = 5,
                   classProbs = TRUE,
                   summaryFunction = twoClassSummary)
# fit the model with formula = out.formula2
fit.cv.bin<-train(out.formula2, trControl = ctrl,
               data = ObsData, method = "bag",
               bagControl = bagControl(fit = ldaBag$fit,
                                       predict = ldaBag$pred,
                                       aggregate = ldaBag$aggregate),
               metric="ROC")
#> Warning: executing %dopar% sequentially: no parallel backend registered
fit.cv.bin
#> Bagged Model
#>
#> 5735 samples
#>   50 predictor
#>    2 classes: 'No', 'Yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 4587, 4589, 4587, 4589, 4588
#> Resampling results:
#>
#>   ROC        Sens       Spec
#>   0.7506666  0.4485809  0.8602811
#>
#> Tuning parameter 'vars' was held constant at a value of 63
```

- Bagging improves prediction accuracy
  - over prediction using a single tree
- Looses interpretability
  - as this is an average of many diagrams now
- But we can get a summary of the importance of each variable

0.0.0.0.1 * Bagging Variable importance

```r
caret::varImp(fit.cv.bin, scale = FALSE)
#> ROC curve variable importance
#>
#>   only 20 most important variables shown (out of 50)
#>
#>                     Importance
#> age                     0.6159
#> APACHE.score            0.6140
#> DASIndex                0.5962
#> Cancer                  0.5878
#> Creatinine              0.5835
#> Tumor                   0.5807
#> blood.pressure          0.5697
#> Glasgow.Coma.Score      0.5656
#> Disease.category        0.5641
#> Temperature             0.5584
#> DNR.status              0.5572
#> Hematocrit              0.5525
#> Weight                  0.5424
#> Bilirubin               0.5397
#> income                  0.5319
#> Immunosupperssion       0.5278
#> RHC.use                 0.5263
#> Dementia                0.5252
#> Congestive.HF           0.5250
#> Hematologic.Diag        0.5250
```

Cross-validation boosting

- Boosting
  - sequentially updated/weighted bootstrap based on previous learning

```r
set.seed(504)
require(caret)
ctrl<-trainControl(method = "cv", number = 5,
                   classProbs = TRUE,
                   summaryFunction = twoClassSummary)
# fit the model with formula = out.formula2
```

```r
fit.cv.bin<-train(out.formula2, trControl = ctrl,
                data = ObsData, method = "gbm",
                verbose = FALSE,
                metric="ROC")
fit.cv.bin
#> Stochastic Gradient Boosting
#>
#> 5735 samples
#>   50 predictor
#>    2 classes: 'No', 'Yes'
#>
#> No pre-processing
#> Resampling: Cross-Validated (5 fold)
#> Summary of sample sizes: 4587, 4589, 4587, 4589, 4588
#> Resampling results across tuning parameters:
#>
#>   interaction.depth  n.trees  ROC        Sens       Spec
#>   1                   50      0.7218938  0.2145970  0.9505647
#>   1                  100      0.7410292  0.2980581  0.9234228
#>   1                  150      0.7483014  0.3487142  0.9030028
#>   2                   50      0.7414513  0.2960631  0.9263816
#>   2                  100      0.7534264  0.3869684  0.8917212
#>   2                  150      0.7575826  0.4187512  0.8777477
#>   3                   50      0.7496078  0.3626125  0.9070358
#>   3                  100      0.7579645  0.4078244  0.8764076
#>   3                  150      0.7637074  0.4445909  0.8702298
#>
#> Tuning parameter 'shrinkage' was held constant at a value of 0.1
#>
#> Tuning parameter 'n.minobsinnode' was held constant at a value of 10
#> ROC was used to select the optimal model using the largest value.
#> The final values used for the model were n.trees = 150, interaction.depth =
#>  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```r
plot(fit.cv.bin)
```

Ensemble methods (Type II)

We introduce the concept of Super Learner, providing external resources for further exploration.

Training different models on the same data

Super Learner

- Large number of candidate learners (CL) with different strengths
  - Parametric (logistic)
  - Non-parametric (CART)
- Cross-validation: CL applied on training data, prediction made on test data
- Final prediction uses a weighted version of all predictions
  - Weights = coef of Observed outcome ~ prediction from each CL

Steps

Refer to this tutorial for steps and examples! Refer to the next chapter for more details.

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

> **Tip**
>
> The following is a brief exercise of super learners in the propensity score context, but we will explore more about this topic in the next chapter.

Unsupervised learning

In this chapter, we will talk about unsupervised learning.

In the initial code chunk, we load a specific library that will be utilized for publishing-related functionality throughout the chapter.

Clustering

Clustering is an unsupervised learning algorithm. These algorithms can classify data into multiple groups. Such classification is based on similarity.

Group characteristics include (to the extent that is possible)

- low inter-class similarity: observation from different clusters would be dissimilar
- high intra-class similarity: observation from the same cluster would be similar

Within-cluster variation will be thus minimized by optimizing within-cluster sum of squares of Euclidean distances (Wikipedia 2023b)

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

K-means

K-means is a very popular clustering algorithm, that partitions the data into k groups.

Algorithm:

- Determine a number k (e.g., could be 3)
- randomly select k subjects in a data. Use these points as staring points (centers or cluster mean) for each cluster.
- By Euclidean distance measure (from the initial centers), try to determine in which cluster the remaining points belong.
- compute new mean value for each cluster.
- based on this new mean, try to determine again in which cluster the data points belong.
- process continues until the data points do not change cluster membership.

Read previously saved data

We read a previously saved dataset from a specified file path.

```
ObsData <- readRDS(file = "Data/machinelearning/rhcAnalytic.RDS")
```

In the next few code chunks, we implement k-means clustering on various subsets of the data, visualizing the results and displaying the cluster centers. The first example uses two variables, the second example uses three, and in the third example, a larger subset of variables is selected but not immediately utilized in the clustering. In the subsequent code chunk, we apply k-means clustering to the larger subset of variables, displaying various results and aggregating data by cluster to display mean and standard deviation values for each variable within each cluster.

Example 1

```
datax0 <- ObsData[c("Heart.rate", "edu")]
kres0 <- kmeans(datax0, centers = 2, nstart = 10)
kres0$centers
#>   Heart.rate      edu
#> 1   54.55138 11.44494
#> 2  134.96277 11.75466
plot(datax0, col = kres0$cluster, main = kres0$tot.withinss)
```

Example 2

```
datax0 <- ObsData[c("blood.pressure", "Heart.rate", "Respiratory.rate")]
kres0 <- kmeans(datax0, centers = 2, nstart = 10)
kres0$centers
#>   blood.pressure Heart.rate Respiratory.rate
#> 1       73.71684   54.95789         22.76723
#> 2       80.10812  135.08956         29.85267
plot(datax0, col = kres0$cluster, main = kres0$tot.withinss)
```

Example with many variables

```r
datax <- ObsData[c("edu", "blood.pressure", "Heart.rate",
                   "Respiratory.rate" , "Temperature",
                   "PH", "Weight", "Length.of.Stay")]
```

```r
kres <- kmeans(datax, centers = 3)
#kres
head(kres$cluster)
#> [1] 1 1 1 3 1 2
kres$size
#> [1] 2793 1688 1254
kres$centers
#>        edu blood.pressure Heart.rate Respiratory.rate Temperature       PH
#> 1 11.85833       54.26924  136.37451         29.76119    37.85078 7.385249
#> 2 11.54214      128.33886  126.12026         29.36611    37.68129 7.401027
#> 3 11.46134       65.47249   53.24242         22.65973    37.01597 7.378482
#>     Weight Length.of.Stay
#> 1 68.63384       23.42356
#> 2 66.68351       20.68128
#> 3 67.57291       18.58931
aggregate(datax, by = list(cluster = kres$cluster), mean)
#>   cluster      edu blood.pressure Heart.rate Respiratory.rate Temperature
#> 1       1 11.85833       54.26924  136.37451         29.76119    37.85078
#> 2       2 11.54214      128.33886  126.12026         29.36611    37.68129
#> 3       3 11.46134       65.47249   53.24242         22.65973    37.01597
#>         PH   Weight Length.of.Stay
#> 1 7.385249 68.63384       23.42356
#> 2 7.401027 66.68351       20.68128
#> 3 7.378482 67.57291       18.58931
aggregate(datax, by = list(cluster = kres$cluster), sd)
#>   cluster      edu blood.pressure Heart.rate Respiratory.rate Temperature
#> 1       1 3.162485       11.93763   23.13140         13.67791    1.781692
#> 2       2 3.091605       18.58070   27.68369         14.08169    1.610746
#> 3       3 3.160538       31.89150   23.63993         13.60831    1.832389
#>          PH   Weight Length.of.Stay
#> 1 0.1082140 27.99506       29.01143
#> 2 0.1009567 32.15078       23.37223
#> 3 0.1226041 26.87075       20.82024
```

Optimal number of clusters

Next, we explore determining the optimal number of clusters, visualizing the total within-cluster sum of squares for different values of k and indicating a chosen value of k with a vertical line on the plot.

```r
require(factoextra)
#> Loading required package: factoextra
#> Loading required package: ggplot2
#> Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
fviz_nbclust(datax, kmeans, method = "wss")+
  geom_vline(xintercept=3,linetype=3)
```

Here the vertical line is chosen based on elbow method (Wikipedia 2023c).

Discussion

- We need to supply a number, k: but we can test different ks to identify optimal value
- Clustering can be influenced by outliners, so median based clustering is possible

- mere ordering can influence clustering, hence we should choose different initial means (e.g., `nstart` should be greater than 1).

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

References

NHIS Example

The tutorial aims to guide the users through fitting machine learning (ML) techniques with health survey data. We will use the National Health Interview Survey (NHIS) 2016 dataset to develop prediction models for predicting high impact chronic pain (HICP) among adults aged 65 years or older. We will use LASSO and random forest models with sampling weights to obtain population-level predictions. In this tutorial, the split-sample approach as an internal validation technique will be used. You can review the earlier tutorial on data splitting technique. Note that this split-sample approach is flagged as a problematic approach in the literature (Ewout W. Steyerberg et al. 2001). The better approach could be cross-validation and bootstrapping Ewout W. Steyerberg and Steyerberg (2019). In the next tutorial, we will apply the ML techniques for survey data with cross-validation.

> **Note**
>
> For those interested in the National Health Interview Survey (NHIS) dataset, can review the earlier tutorial about the dataset.

Steyerberg EW, Harrell Jr FE, Borsboom GJ, Eijkemans MJ, Vergouwe Y, Habbema JD. Internal validation of predictive models: efficiency of some procedures for logistic regression analysis. Journal of Clinical Epidemiology. 2001; 54(8):774-81. DOI: 10.1016/S0895-4356(01)00341-9

Steyerberg EW, Steyerberg EW. Overfitting and optimism in prediction models. Clinical prediction models: A practical approach to development, validation, and updating. 2019:95-112. DOI: 10.1007/978-3-030-16399-0_5

Load packages

We load several R packages required for fitting LASSO and random forest models.

```r
# Load required packages
library(tableone)
library(gtsummary)
library(glmnet)
```

```r
library(WeightedROC)
library(ranger)
library(scoring)
library(DescTools)
library(ggplot2)
library(mlr3misc)
```

Analytic dataset

Load

We load the dataset into the R environment and lists all available variables and objects.

```r
load("Data/machinelearning/nhis2016.RData")
ls()
#> [1] "dat.analytic"    "has_annotations"
```

```r
dim(dat.analytic)
#> [1] 7828   14

head(dat.analytic)
#>           studyid psu strata weight HICP    sex             marital   race
#> 3  20160000020101   2    149   1717    0   Male   Divorced/separated  White
#> 11 20160000250101   2    117   4088    0   Male Married/with partner Others
#> 12 20160000260102   5    149   2566    0   Male Married/with partner  White
#> 13 20160000290101  57    100   1311    1   Male   Divorced/separated  White
#> 16 20160000350101  26    136   2903    0 Female   Divorced/separated  Black
#> 28 20160000740101  36    103   1278    0 Female             Widowed  White
#>    poverty.status diabetes high.cholesterol stroke arthritis current.smoker
#> 3    200-400% FPL       No               Yes     No        No             No
#> 11      400%+ FPL       No               Yes     No        No             No
#> 12   200-400% FPL       No                No     No       Yes             No
#> 13      400%+ FPL       No                No     No       Yes             No
#> 16   200-400% FPL       No               Yes     No       Yes             No
#> 28   200-400% FPL       No                No     No       Yes             No
```

The dataset contains 7,828 complete case participants (i.e., no missing) with 14 variables:

- `studyid`: Unique identifier
- `psu`: Pseudo-PSU
- `strata`: Pseudo-stratum
- `weight`: Sampling weight
- `HICP`: HICP (high impact chronic pain, the binary outcome variable)
- `sex`: Sex
- `marital`: Marital status
- `race`: Race/ethnicity
- `poverty.status`: Poverty status
- `diabetes`: Diabetes
- `high.cholesterol`: High cholesterol
- `stroke`: Stroke
- `arthritis`: Arthritis and rheumatism
- `current.smoker`: Current smoker

Let's see the descriptive statistics of the predictors stratified by the outcome variable (HICP).

Descriptive statistics

```r
# Predictors
predictors <- c("sex", "marital", "race", "poverty.status",
                "diabetes", "high.cholesterol", "stroke",
                "arthritis", "current.smoker")

# Table 1 - Unweighted
#tab1 <- CreateTableOne(vars = predictors, strata = "HICP",
#                       data = dat.analytic, test = F)
#print(tab1, showAllLevels = T)

tbl_summary(data = dat.analytic, include = predictors,
            by = HICP, missing = "no") %>%
  modify_spanning_header(c("stat_1", "stat_2") ~ "**HICP**")
```

Characteristic	**0**, N = 6,847	**1**, N = 981
sex		
Female	3,841 (56%)	623 (64%)

Characteristic	**0**, N = 6,847	**1**, N = 981
Male	3,006 (44%)	358 (36%)
marital		
Never married	444 (6.5%)	60 (6.1%)
Married/with partner	3,183 (46%)	379 (39%)
Divorced/separated	1,252 (18%)	198 (20%)
Widowed	1,968 (29%)	344 (35%)
race		
White	5,455 (80%)	756 (77%)
Black	606 (8.9%)	99 (10%)
Hispanic	431 (6.3%)	79 (8.1%)
Others	355 (5.2%)	47 (4.8%)
poverty.status		
<100% FPL	543 (7.9%)	176 (18%)
100-200% FPL	1,520 (22%)	307 (31%)
200-400% FPL	2,309 (34%)	309 (31%)
400%+ FPL	2,475 (36%)	189 (19%)
diabetes	1,275 (19%)	315 (32%)
high.cholesterol	3,602 (53%)	633 (65%)
stroke	527 (7.7%)	146 (15%)
arthritis	3,226 (47%)	769 (78%)
current.smoker	619 (9.0%)	123 (13%)

Weight normalization

Now, we will fit the LASSO model for predicting binary HICP with the listed predictors. Note that we are not interested in the statistical significance of the β coefficients. Hence, not utilizing PSU and strata should not be an issue in this prediction problem. However, we still need to use sampling weights to get population-level predictions. Large weights are usually problematic, particularly with model evaluation. One way to solve the problem is weight normalization (Bruin 2023).

```r
# Normalize weight
dat.analytic$wgt <- dat.analytic$weight *
  nrow(dat.analytic)/sum(dat.analytic$weight)

# Weight summary
```

```r
summary(dat.analytic$weight)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>     243    1521    2604    2914    3791   14662
summary(dat.analytic$wgt)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> 0.08339 0.52198 0.89365 1.00000 1.30109 5.03175

# The weighted and unweighted n are equal
nrow(dat.analytic)
#> [1] 7828
sum(dat.analytic$wgt)
#> [1] 7828
```

Split-sample

Let us create our training and test data using the split-sample approach. We created 70% training and 30% test data for our example.

```r
set.seed(604001)
dat.analytic$datasplit <- rbinom(nrow(dat.analytic),
                                 size = 1, prob = 0.7)
table(dat.analytic$datasplit)
#>
#>    0    1
#> 2343 5485

# Training data
dat.train <- dat.analytic[dat.analytic$datasplit == 1,]
dim(dat.train)
#> [1] 5485   16

# Test data
dat.test <- dat.analytic[dat.analytic$datasplit == 0,]
dim(dat.test)
#> [1] 2343   16
```

Regression formula

Let's us define the regression formula:

```
Formula <- formula(paste("HICP ~ ", paste(predictors,
                                          collapse=" + ")))
Formula
#> HICP ~ sex + marital + race + poverty.status + diabetes + high.cholesterol +
#>     stroke + arthritis + current.smoker
```

LASSO for Surveys

Now, we will fit the LASSO model for our survey data. Here are the steps:

- We will fit 5-fold cross-validation on the training data to find the value of lambda that gives minimum prediction error. We will incorporate sampling weights in the model to account for survey data.
- Fit LASSO on the training with the optimum lambda from the previous step. Incorporate sampling weights in the model to account for survey data.
- Calculate predictive performance (e.g., AUC) on the test data.

Data in matrix

To perform LASSO with the `glmnet` package, we need to set the predictors in the data.matrix format and outcome variable as a vector.

```
# Training data - X: predictor, y: outcome
X.train <- model.matrix(Formula, dat.train)[,-1]
y.train <- as.matrix(dat.train$HICP)

# Test data - X: predictor, y: outcome
X.test <- model.matrix(Formula, dat.test)[,-1]
y.test <- as.matrix(dat.test$HICP)
```

Let us see the few rows of the data:

```r
head(X.train)
#>    sexMale maritalMarried/with partner maritalDivorced/separated maritalWidowed
#> 12       1                           1                          0              0
#> 13       1                           0                          1              0
#> 16       0                           0                          1              0
#> 42       0                           0                          0              1
#> 63       0                           0                          0              1
#> 65       0                           0                          0              1
#>    raceBlack raceHispanic raceOthers poverty.status100-200% FPL
#> 12         0            0          0                          0
#> 13         0            0          0                          0
#> 16         1            0          0                          0
#> 42         0            0          0                          0
#> 63         0            0          0                          0
#> 65         0            1          0                          1
#>    poverty.status200-400% FPL poverty.status400%+ FPL diabetesYes
#> 12                          1                       0           0
#> 13                          0                       1           0
#> 16                          1                       0           0
#> 42                          0                       1           0
#> 63                          1                       0           0
#> 65                          0                       0           1
#>    high.cholesterolYes strokeYes arthritisYes current.smokerYes
#> 12                   0         0            1                  0
#> 13                   0         0            1                  0
#> 16                   1         0            1                  0
#> 42                   1         0            0                  0
#> 63                   0         0            0                  0
#> 65                   1         0            1                  0
```

As we can see, factor predictors are coded into dummy variables. It is important to note that the continuous predictors should be standardized. glmnet does this by default. Next, we will use the `glmnet` function to fit the LASSO model.

> **Note**
>
> In glmnet function, `alpha = 1` for the LASSO, `alpha = 0` for the ridge, and setting alpha to some value between 0 and 1 is the elastic net model.

Find best lambda

Now, we will use k-fold cross-validation with the `cv.glmnet` function to find the best lambda value. In this example, we choose k = 5. Note that we must incorporate sampling weight to account for survey data.

```
# Find the best lambda using 5-fold CV
fit.cv.lasso <- cv.glmnet(x = X.train, y = y.train,
                          nfolds = 5, alpha = 1,
                          family = "binomial",
                          weights = dat.train$wgt)
fit.cv.lasso
#>
#> Call:  cv.glmnet(x = X.train, y = y.train, weights = dat.train$wgt,      nfolds = 5, alpha =
#>
#> Measure: Binomial Deviance
#>
#>       Lambda Index Measure      SE Nonzero
#> min 0.001355    43  0.6856 0.01905      12
#> 1se 0.020128    14  0.7036 0.01446       5
```

We can also plot all the lambda values against the deviance (i.e., prediction error).

```
plot(fit.cv.lasso)
```

```
# Best lambda
fit.cv.lasso$lambda.min
#> [1] 0.001355482
```

The lambda value that has the lowest deviance is 0.0013555. Our next step is to fit the LASSO model with the best lambda. Again, we must incorporate sampling weight to account for survey data.

LASSO with best lambda

```
# Fit the model on the training set with optimum lambda
fit.lasso <- glmnet(x = X.train, y = y.train,
                    alpha = 1, family = "binomial",
                    lambda = fit.cv.lasso$lambda.min,
                    weights = dat.train$wgt)
fit.lasso
#>
#> Call:  glmnet(x = X.train, y = y.train, family = "binomial", weights = dat.train$wgt,
#>
#>   Df %Dev   Lambda
#> 1 12 9.68 0.001355
```

Let's check the coefficients from the model:

```r
# Intercept
fit.lasso$a0
#>        s0
#> -2.491484

# Beta coefficients
fit.lasso$beta
#> 15 x 1 sparse Matrix of class "dgCMatrix"
#>                                       s0
#> sexMale                      -0.009832818
#> maritalMarried/with partner   .
#> maritalDivorced/separated     .
#> maritalWidowed                0.041111308
#> raceBlack                    -0.064853067
#> raceHispanic                 -0.037985160
#> raceOthers                    .
#> poverty.status100-200% FPL  -0.268494032
#> poverty.status200-400% FPL  -0.602097419
#> poverty.status400%+ FPL     -1.052635980
#> diabetesYes                   0.369667918
#> high.cholesterolYes           0.299475863
#> strokeYes                     0.449050679
#> arthritisYes                  1.241288036
#> current.smokerYes             0.253439388
```

As we can see, the coefficient is not shown for some predictors. This is because the LASSO model shrunk the coefficient to zero. In other words, these predictors were dropped entirely from the model because they were not contributing enough to predict the outcome. next, we will use the final model to make predictions on new observations or our test data.

```r
# Pr.(HICP = Yes) on the test set
dat.test$pred.lasso <- predict(fit.lasso,
                               newx = X.test,
                               type = "response")
head(dat.test$pred.lasso)
#>            s0
#> 3  0.05711170
#> 11 0.03716633
```

```
#> 28 0.14049527
#> 30 0.05764351
#> 31 0.11408034
#> 59 0.32540267
```

Model performance

Now, we will calculate the model performance measures such as AUC, calibration slope, and Brier score Christodoulou et al. (2019). We will incorporate sampling weights to get population-level estimates.

> **Note**
>
> - Area under the curve (AUC) is a measure of discrimination or accuracy of a model. A higher AUC is better. An AUC value of 1 is considered a perfect prediction, while an AUC value of 0.50 is no better than a coin toss. In practice, AUC values of 0.70 to 0.80 are considered good, and those > 0.80 are considered very good.
> - Calibration is defined as the agreement between observed and predicted probability of the outcome. In this exercise, we will estimate the calibration slope as a measure of calibration. A calibration slope of 1 reflects a well-calibrated model, a calibration slope less than 1 indicates overfitting and greater than 1 indicates underfitting of the model.
> - The Brier score is a measure of overall performance. The Brier score can range from 0 to 1 and is similar to the mean squared error. A lower Brier score value (closer to 0) indicates a better model.

Steyerberg EW, Vickers AJ, Cook NR, Gerds T, Gonen M, Obuchowski N, Pencina MJ, Kattan MW. Assessing the performance of prediction models: a framework for some traditional and novel measures. Epidemiology (Cambridge, Mass.). 2010;21(1):128. DOI: 10.1097/EDE.0b013e3181c30fb2

Steyerberg EW, Vergouwe Y. Towards better clinical prediction models: seven steps for development and an ABCD for validation. European heart journal. 2014;35(29):1925-31. DOI: 10.1093/eurheartj/ehu207

Christodoulou E, Ma J, Collins GS, Steyerberg EW, Verbakel JY, Van Calster B. A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models. Journal of clinical epidemiology. 2019;110:12-22. DOI: 10.1016/j.jclinepi.2019.02.004


```r
# AUC on the test set with sampling weights
auc.lasso <- WeightedAUC(WeightedROC(dat.test$pred.lasso,
                                     dat.test$HICP,
                                     weight = dat.test$wgt))
```

```r
auc.lasso
#> [1] 0.7662941
```

```r
# Logit of the predicted probability
dat.test$pred.lasso.logit <- Logit(dat.test$pred.lasso)

# Weighted calibration slope
mod.cal <- glm(HICP ~ pred.lasso.logit, data = dat.test,
               family = binomial, weights = wgt)
cal.slope.lasso <- summary(mod.cal)$coef[2,1]
cal.slope.lasso
#> [1] 1.244645
```

```r
# Weighted Brier Score
brier.lasso <- mean(brierscore(HICP ~ dat.test$pred.lasso,
                               data = dat.test,
                               wt = dat.test$wgt))
brier.lasso
#> [1] 0.09978551
```

Random Forest for Surveys

Now, we will fit the random forest model for predicting binary HICP with the listed predictors. Here are the steps for fitting the model:

- Fit random forest model on the training set to find the value of the hyperparameters (number of trees, number of predictors to split at in each node, and minimal node size to split at) that gives minimum prediction error. Incorporate sampling weights in the model to account for survey data.
- Grid-search with out-of-sample error approach is widely used in the literature. In this approach, we create a data frame from all combinations of the hyperparameters and check which combination gives the lowest out-of-sample error.

- Fit the random forest model on the training with the selected hyperparameters from the previous step. Incorporate sampling weights in the model to account for survey data.
- Calculate predictive performance (e.g., AUC) on the test data.

Formula

We will use the same formula defined above.

```
Formula
#> HICP ~ sex + marital + race + poverty.status + diabetes + high.cholesterol +
#>     stroke + arthritis + current.smoker
```

Hyperparameter tuning

For tuning the hyperparameters, let's use the grid search approach.

```
# Grid with 1000 models - huge time consuming
#grid.search <- expand.grid(mtry = 1:10, node.size = 1:10,
#                            num.trees = seq(50,500,50),
#                            oob_error = 0)

# Grid with 36 models as an exercise
grid.search <- expand.grid(mtry = 5:7, node.size = 1:3,
                           num.trees = seq(200,500,100),
                           oob_error = 0)
head(grid.search)
#>   mtry node.size num.trees oob_error
#> 1    5         1       200         0
#> 2    6         1       200         0
#> 3    7         1       200         0
#> 4    5         2       200         0
#> 5    6         2       200         0
#> 6    7         2       200         0
```

Now, we will fit the random forest model with the selected grids. We will incorporate sampling weight as the case weight in the `ranger` function.

```
## Calculate prediction error for each grid
for(ii in 1:nrow(grid.search)) {
  # Model on training set with grid
  fit.rf.tune <- ranger(formula = Formula,
                        data = dat.train,
                        num.trees = grid.search$num.trees[ii],
                        mtry = grid.search$mtry[ii],
                        min.node.size = grid.search$node.size[ii],
                        importance = 'impurity',
                        case.weights = dat.train$wgt)

  # Add Out-of-bag (OOB) error to each grid
  grid.search$oob_error[ii] <- sqrt(fit.rf.tune$prediction.error)
}
head(grid.search)
#>   mtry node.size num.trees oob_error
#> 1    5         1       200 0.3343642
#> 2    6         1       200 0.3377220
#> 3    7         1       200 0.3404995
#> 4    5         2       200 0.3342145
#> 5    6         2       200 0.3375921
#> 6    7         2       200 0.3410897
```

Let's check which combination of hyperparameters (number of trees, number of predictors to split at in each node, and minimal node size to split at) gives minimum prediction error.

```
position <- which.min(grid.search$oob_error)
grid.search[position,]
#>    mtry node.size num.trees oob_error
#> 16    5         3       300 0.3327845
```

Model after tuning

Now, we will fit the random forest model with the tuned hyperparameters.

```r
# Fit the model on the training set
fit.rf <- ranger(formula = Formula,
                 data = dat.train,
                 case.weights = dat.train$wgt,
                 probability = T,
                 num.trees = grid.search$num.trees[position],
                 min.node.size = grid.search$node.size[position],
                 mtry = grid.search$mtry[position],
                 importance = 'impurity')

# Fitted random forest model
fit.rf
#> Ranger result
#>
#> Call:
#>  ranger(formula = Formula, data = dat.train, case.weights = dat.train$wgt,      probability
#>
#> Type:                             Probability estimation
#> Number of trees:                  300
#> Sample size:                      5485
#> Number of independent variables:  9
#> Mtry:                             5
#> Target node size:                 3
#> Variable importance mode:         impurity
#> Splitrule:                        gini
#> OOB prediction error (Brier s.):  0.1110941
```

Now, we can use the model to make predictions on our test data.

```r
# Pr.(HICP = Yes) on the test set
dat.test$pred.rf <- predict(fit.rf,
                            data = dat.test)$predictions[,2]
head(dat.test$pred.rf)
#> [1] 0.031093101 0.003712312 0.129992168 0.044802288 0.025764960 0.307125058
```

Model performance

The same as the LASSO model, we can calculate the AUC, calibration slope, and Brier score.

```r
# AUC on the test set with sampling weights
auc.rf <- WeightedAUC(WeightedROC(dat.test$pred.rf,
                                  dat.test$HICP,
                                  weight = dat.test$wgt))
auc.rf
#> [1] 0.6941022
```

```r
# Logit of the predicted probability
dat.test$pred.rf[dat.test$pred.rf == 0] <- 0.00001
dat.test$pred.rf.logit <- Logit(dat.test$pred.rf)

# Weighted calibration slope
mod.cal <- glm(HICP ~ pred.rf.logit,
               data = dat.test,
               family = binomial,
               weights = wgt)
cal.slope.rf <- summary(mod.cal)$coef[2,1]
cal.slope.rf
#> [1] 0.4977901
```

```r
# Weighted Brier Score
brier.rf <- mean(brierscore(HICP ~ dat.test$pred.rf,
                            data = dat.test,
                            wt = dat.test$wgt))
brier.rf
#> [1] 0.1095384
```

Variable importance

One nice feature of random forest is that we can rank the variables and generate a variable importance plot.

```r
ggplot(
  enframe(fit.rf$variable.importance,
          name = "variable",
          value = "importance"),
  aes(x = reorder(variable, importance),
      y = importance, fill = importance)) +
  geom_bar(stat = "identity",
```

```
          position = "dodge") +
  coord_flip() +
  ylab("Variable Importance") +
  xlab("") +
  ggtitle("") +
  guides(fill = "none") +
  scale_fill_gradient(low = "grey",
                      high = "grey10") +
  theme_bw()
```

As per the figure, marital status, poverty status, sex, and arthritis are the most influential predictors in predicting HICP, while stroke is the least important predictor.

Performance comparison

Model	AUC	Calibration slope	Brier score
LASSO	0.7662941	1.2446448	0.0997855
Random forest	0.6941022	0.4977901	0.1095384

References

Replicate Results

The tutorial aims to guide the users through fitting machine learning techniques with health survey data. We will replicate some of the results of this article by Falasinnu et al. (2023).

The authors used the National Health Interview Survey (NHIS) 2016 dataset to develop prediction models for predicting high impact chronic pain (HICP). They also evaluated the predictive performances of the models within sociodemographic subgroups, such as sex (male, female), age (< 65, ≥ 65), and race/ethnicity (White, Black, Hispanic). They used LASSO and random forest models with 5-fold cross-validation as an internal validation. To obtain population-level predictions, they account for survey weights in both models.

Falasinnu T, Hossain MB, Weber II KA, Helmick CG, Karim ME, Mackey S. The Problem of Pain in the United States: A Population-Based Characterization of Biopsychosocial Correlates of High Impact Chronic Pain Using the National Health Interview Survey. The Journal of Pain. 2023;24(6):1094-103. DOI: 10.1016/j.jpain.2023.03.008

For those interested in the National Health Interview Survey (NHIS) dataset, can review the earlier tutorial about the dataset.

> **Note**
>
> To handle missing data in the predictors, they used multiple imputation technique. However, for simplicity, this tutorial focuses on a complete case dataset. We will also only focus on predicting HICP for people aged 65 years or older (a dataset of ~8,800 participants compared to the dataset of 33,000 participants aged 18 years or older).

Load packages

We load several R packages required for fitting LASSO and random forest models.

```r
# Load required packages
library(tableone)
library(gtsummary)
```

```r
library(glmnet)
library(WeightedROC)
library(ranger)
library(scoring)
library(DescTools)
library(ggplot2)
library(mlr3misc)
```

Analytic dataset

Load

We load the dataset into the R environment and lists all available variables and objects.

```r
load("Data/machinelearning/Falasinnu2023.RData")
ls()
#> [1] "dat"              "has_annotations"
```

```r
dim(dat)
#> [1] 8881   49
```

The dataset contains 8,881 participants aged 65 years or older with 49 variables:

- `studyid`: Unique identifier
- `psu`: Pseudo-PSU
- `strata`: Pseudo-stratum
- `weight`: Sampling weight
- `HICP`: HICP (binary outcome variable)
- `age`: Age
- `sex`: Sex
- `hhsize`: Number of people in household
- `born`: Citizenship
- `marital`: Marital status
- `region`: Region
- `race`: Race/ethnicity
- `education`: Education
- `employment.status`: Employment status

- `poverty.status`: Poverty status
- `veteran`: Veteran
- `insurance`: Health insurance coverage
- `sex.orientation`: Sexual orientation
- `worried.money`: Worried about money
- `good.neighborhood`: Good neighborhood
- `psy.symptom`: Psychological symptoms
- `visit.ED`: Number of times in ER/ED
- `surgery`: Number of surgeries in past 12 months
- `dr.visit`: Time since doctor visits
- `cancer`: Cancer
- `asthma`: Asthma
- `htn`: Hypertension
- `liver.disease`: Liver disease
- `diabetes`: Diabetes
- `ulcer`: Ulcer
- `stroke`: Stroke
- `emphysema`: Emphysema
- `copd`: COPD
- `high.cholesterol`: High cholesterol
- `coronary.heart.disease`: Coronary heart disease
- `angina`: Angina pectoris
- `heart.attack`: Heart attack
- `heart.disease`: Heart condition/disease
- `arthritis`: Arthritis and rheumatism
- `crohns.disease`: Crohn's disease
- `place.routine.care`: Usual place for routine care
- `trouble.asleep`: Trouble falling asleep
- `obese`: Obesity
- `current.smoker`: Current smoker
- `heavy.drinker`: Heavy drinker
- `hospitalization`: Hospital stay days
- `better.health.status`: Better health status
- `physical.activity`: Physical activity

See the NHIS 2016 dataset and the article for better understanding of the variables.

Complete case data

```r
# Age
table(dat$age, useNA = "always")
#>
#>  <65  65+ <NA>
#>    0 8881    0
```

Let us consider a complete case dataset

```r
dat.complete <- na.omit(dat)
dim(dat.complete)
#> [1] 7280   49
```

As we can see, there are 7,280 participants with complete case information. Let's see the descriptive statistics of the predictors stratified by HICP.

Descriptive statistics

```r
# Predictors
predictors <- c("sex", "hhsize", "born", "marital",
                "region", "race", "education",
                "employment.status", "poverty.status",
                "veteran", "insurance",
                "sex.orientation", "worried.money",
                "good.neighborhood",
                "psy.symptom", "visit.ED", "surgery",
                "dr.visit", "cancer",
                "asthma", "htn", "liver.disease",
                "diabetes", "ulcer", "stroke",
                "emphysema", "copd", "high.cholesterol",
                "coronary.heart.disease",
                "angina", "heart.attack",
                "heart.disease", "arthritis",
                "crohns.disease", "place.routine.care",
                "trouble.asleep", "obese",
```

```r
                "current.smoker", "heavy.drinker",
                "hospitalization",
                "better.health.status",
                "physical.activity")

# Table 1 - Unweighted
tbl_summary(data = dat.complete,
            include = predictors,
            by = HICP, missing = "no") %>%
  modify_spanning_header(c("stat_1",
                           "stat_2") ~ "**HICP**")
```

Characteristic	**0**, N = 6,389	**1**, N = 891
sex		
Female	3,587 (56%)	569 (64%)
Male	2,802 (44%)	322 (36%)
hhsize	2 (1, 2)	2 (1, 2)
born		
Born in US	5,775 (90%)	802 (90%)
Other place	614 (9.6%)	89 (10.0%)
marital		
Never married	411 (6.4%)	57 (6.4%)
Married/with partner	2,990 (47%)	349 (39%)
Divorced/separated	1,161 (18%)	179 (20%)
Widowed	1,827 (29%)	306 (34%)
region		
Northeast	1,172 (18%)	143 (16%)
Midwest	1,451 (23%)	189 (21%)
South	2,203 (34%)	331 (37%)
West	1,563 (24%)	228 (26%)
race		
White	5,090 (80%)	694 (78%)
Black	564 (8.8%)	88 (9.9%)
Hispanic	406 (6.4%)	70 (7.9%)
Others	329 (5.1%)	39 (4.4%)
education		
Less than high school	954 (15%)	220 (25%)
High school/GED	1,863 (29%)	269 (30%)
Some college	1,716 (27%)	248 (28%)

Characteristic	**0**, N = 6,389	**1**, N = 891
Bachelors degree or higher	1,856 (29%)	154 (17%)
employment.status		
Employed hourly	578 (9.0%)	22 (2.5%)
Employed non-hourly	608 (9.5%)	27 (3.0%)
Worked previously	4,923 (77%)	777 (87%)
Never worked	280 (4.4%)	65 (7.3%)
poverty.status		
<100% FPL	496 (7.8%)	154 (17%)
100-200% FPL	1,401 (22%)	276 (31%)
200-400% FPL	2,157 (34%)	283 (32%)
400%+ FPL	2,335 (37%)	178 (20%)
veteran	1,482 (23%)	163 (18%)
insurance		
Uninsured	32 (0.5%)	6 (0.7%)
Medicaid/Medicare	2,995 (47%)	478 (54%)
Privately Insured	2,847 (45%)	311 (35%)
Other	515 (8.1%)	96 (11%)
sex.orientation		
Heterosexual	6,224 (97%)	861 (97%)
Other	165 (2.6%)	30 (3.4%)
worried.money	2,351 (37%)	491 (55%)
good.neighborhood	5,988 (94%)	781 (88%)
psy.symptom	694 (11%)	353 (40%)
visit.ED		
None	5,050 (79%)	531 (60%)
One	949 (15%)	187 (21%)
2-3	313 (4.9%)	123 (14%)
4+	77 (1.2%)	50 (5.6%)
surgery		
None	5,218 (82%)	650 (73%)
One	898 (14%)	176 (20%)
Two	206 (3.2%)	44 (4.9%)
3+	67 (1.0%)	21 (2.4%)
dr.visit		
<6 months	5,390 (84%)	837 (94%)
6-12 months	591 (9.3%)	41 (4.6%)
1-5 years	281 (4.4%)	10 (1.1%)
>5 years/never	127 (2.0%)	3 (0.3%)
cancer	1,566 (25%)	271 (30%)

Characteristic	**0**, N = 6,389	**1**, N = 891
asthma	645 (10%)	176 (20%)
htn	3,953 (62%)	689 (77%)
liver.disease	122 (1.9%)	50 (5.6%)
diabetes	1,179 (18%)	278 (31%)
ulcer	545 (8.5%)	161 (18%)
stroke	485 (7.6%)	130 (15%)
emphysema	235 (3.7%)	76 (8.5%)
copd	496 (7.8%)	145 (16%)
high.cholesterol	3,373 (53%)	577 (65%)
coronary.heart.disease	814 (13%)	211 (24%)
angina	298 (4.7%)	100 (11%)
heart.attack	552 (8.6%)	154 (17%)
heart.disease	1,083 (17%)	210 (24%)
arthritis	3,017 (47%)	700 (79%)
crohns.disease	102 (1.6%)	29 (3.3%)
place.routine.care		
No place	235 (3.7%)	26 (2.9%)
Doctor's office	4,634 (73%)	621 (70%)
Hospital/Clinic	1,403 (22%)	221 (25%)
Other place	117 (1.8%)	23 (2.6%)
trouble.asleep	1,970 (31%)	424 (48%)
obese	1,757 (28%)	397 (45%)
current.smoker	565 (8.8%)	111 (12%)
heavy.drinker	308 (4.8%)	25 (2.8%)
hospitalization		
None	5,009 (78%)	503 (56%)
1-2 days	592 (9.3%)	57 (6.4%)
3-5 days	360 (5.6%)	63 (7.1%)
6+ days	428 (6.7%)	268 (30%)
better.health.status	890 (14%)	119 (13%)
physical.activity		
Less	3,734 (58%)	743 (83%)
Moderate	1,794 (28%)	108 (12%)
High	861 (13%)	40 (4.5%)

LASSO for surveys

Now, we will fit the LASSO model for predicting binary HICP with the listed predictors. Similar to the previous chapter, we will normalize the weight.

Weight normalization

```r
# Normalize weight
dat.complete$wgt <- dat.complete$weight *
  nrow(dat.complete)/sum(dat.complete$weight)

# Weight summary
summary(dat.complete$weight)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>     243    1503    2583    2886    3747   14662
summary(dat.complete$wgt)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.0842  0.5208  0.8950  1.0000  1.2983  5.0804

# The weighted and unweighted n are equal
nrow(dat.complete)
#> [1] 7280
sum(dat.complete$wgt)
#> [1] 7280
```

Folds

Let's create five random folds and specify the regression formula.

```r
k <- 5
set.seed(604)
nfolds <- sample(1:k,
                 size = nrow(dat.complete),
                 replace = T)
table(nfolds)
#> nfolds
```

```
#>    1    2    3    4    5
#> 1451 1457 1496 1468 1408
```

Formula

```r
Formula <- formula(paste("HICP ~ ", paste(predictors,
                                          collapse=" + ")))
Formula
#> HICP ~ sex + hhsize + born + marital + region + race + education +
#>     employment.status + poverty.status + veteran + insurance +
#>     sex.orientation + worried.money + good.neighborhood + psy.symptom +
#>     visit.ED + surgery + dr.visit + cancer + asthma + htn + liver.disease +
#>     diabetes + ulcer + stroke + emphysema + copd + high.cholesterol +
#>     coronary.heart.disease + angina + heart.attack + heart.disease +
#>     arthritis + crohns.disease + place.routine.care + trouble.asleep +
#>     obese + current.smoker + heavy.drinker + hospitalization +
#>     better.health.status + physical.activity
```

5-fold CV LASSO

Now, we will fit the LASSO model with 5-fold cross-validation (CV). Here are the steps:

- For fold 1, folds 2-5 is the training set and fold 1 is the test set
- Fit 5-fold cross-validation on the training set to find the value of lambda that gives minimum prediction error. Incorporate sampling weights in the model to account for survey design.
- Fit LASSO on the training with the optimum lambda from the previous step. Incorporate sampling weights in the model to account for survey design.
- Calculate predictive performance (e.g., AUC) on the test set
- Repeat the analysis for all folds.

```r
fit.lasso <- list(NULL)
auc.lasso <- NULL
cal.slope.lasso <- NULL
brier.lasso <- NULL
for (fold in 1:k) {
  # Training data
  dat.train <- dat.complete[nfolds != fold, ]
  X.train <- model.matrix(Formula, dat.train)[,-1]
  y.train <- as.matrix(dat.train$HICP)

  # Test data
  dat.test <- dat.complete[nfolds == fold, ]
  X.test <- model.matrix(Formula, dat.test)[,-1]
  y.test <- as.matrix(dat.test$HICP)

  # Find the optimum lambda using 5-fold CV
  fit.cv.lasso <- cv.glmnet(x = X.train,
                            y = y.train,
                            nfolds = 5,
                            alpha = 1,
                            family = "binomial",
                            weights = dat.train$wgt)

  # Fit the model on the training set with optimum lambda
  fit.lasso[[fold]] <- glmnet(
    x = X.train,
    y = y.train,
    alpha = 1,
    family = "binomial",
    lambda = fit.cv.lasso$lambda.min,
    weights = dat.train$wgt)

  # Prediction on the test set
  dat.test$pred.lasso <- predict(fit.lasso[[fold]],
                                 newx = X.test,
                                 type = "response")

  # AUC on the test set with sampling weights
  auc.lasso[fold] <- WeightedAUC(
    WeightedROC(dat.test$pred.lasso,
```

```r
                 dat.test$HICP,
                 weight = dat.test$wgt))

  # Weighted calibration slope
  mod.cal <- glm(
    HICP ~ Logit(dat.test$pred.lasso),
    data = dat.test,
    family = binomial,
    weights = wgt)
  cal.slope.lasso[fold] <- summary(mod.cal)$coef[2,1]

  # Weighted Brier Score
  brier.lasso[fold] <- mean(
    brierscore(HICP ~ dat.test$pred.lasso,
               data = dat.test,
               wt = dat.test$wgt))
}
```

Model performance

Let's check how prediction worked.

```r
# Fitted LASSO models
fit.lasso[[1]]
#>
#> Call:  glmnet(x = X.train, y = y.train, family = "binomial", weights = dat.train$wgt,
#>
#>   Df  %Dev   Lambda
#> 1 46 23.91 0.002972
fit.lasso[[2]]
#>
#> Call:  glmnet(x = X.train, y = y.train, family = "binomial", weights = dat.train$wgt,
#>
#>   Df  %Dev   Lambda
#> 1 47 25.32 0.002559
fit.lasso[[3]]
#>
#> Call:  glmnet(x = X.train, y = y.train, family = "binomial", weights = dat.train$wgt,
#>
```

```
#>   Df %Dev   Lambda
#> 1 47 24.3 0.002724
fit.lasso[[4]]
#>
#> Call:  glmnet(x = X.train, y = y.train, family = "binomial", weights = dat.train$wgt,
#>
#>   Df  %Dev   Lambda
#> 1 34 25.27 0.004726
fit.lasso[[5]]
#>
#> Call:  glmnet(x = X.train, y = y.train, family = "binomial", weights = dat.train$wgt,
#>
#>   Df  %Dev   Lambda
#> 1 39 24.32 0.003525
```

```
# Intercept from the LASSO models in different folds
fit.lasso[[1]]$a0
#>        s0
#> -3.733405
fit.lasso[[2]]$a0
#>        s0
#> -3.534898
fit.lasso[[3]]$a0
#>        s0
#> -3.486223
fit.lasso[[4]]$a0
#>        s0
#> -3.800206
fit.lasso[[5]]$a0
#>       s0
#> -3.68776
```

```
# Beta coefficients from the LASSO models in different folds
fit.lasso[[1]]$beta
#> 67 x 1 sparse Matrix of class "dgCMatrix"
#>                                                 s0
#> sexMale                                .
#> hhsize                                 0.0092060605
#> bornOther place                        .
#> maritalMarried/with partner            .
```

```
#> maritalDivorced/separated              .
#> maritalWidowed                         0.0926365970
#> regionMidwest                          .
#> regionSouth                            .
#> regionWest                             0.1470219542
#> raceBlack                             -0.0436983700
#> raceHispanic                           .
#> raceOthers                             .
#> educationHigh school/GED               .
#> educationSome college                  .
#> educationBachelors degree or higher    .
#> employment.statusEmployed non-hourly   .
#> employment.statusWorked previously     0.5412332930
#> employment.statusNever worked          0.8300536966
#> poverty.status100-200% FPL             0.0636289868
#> poverty.status200-400% FPL            -0.0697612513
#> poverty.status400%+ FPL               -0.2887583235
#> veteranYes                            -0.0300714331
#> insuranceMedicaid/Medicare             .
#> insurancePrivately Insured            -0.0924641963
#> insuranceOther                         0.1503323815
#> sex.orientationOther                   0.0765093892
#> worried.moneyYes                       0.2812674935
#> good.neighborhoodYes                  -0.2491796538
#> psy.symptomYes                         0.9726200151
#> visit.EDOne                            0.0674459188
#> visit.ED2-3                            0.1375781535
#> visit.ED4+                             0.4225671575
#> surgeryOne                             .
#> surgeryTwo                             .
#> surgery3+                             -0.0839622285
#> dr.visit6-12 months                   -0.2120063637
#> dr.visit1-5 years                     -0.4068474570
#> dr.visit>5 years/never                -0.1453128227
#> cancerYes                              0.0993613856
#> asthmaYes                              0.2997325771
#> htnYes                                 0.2082877598
#> liver.diseaseYes                       0.8058966864
#> diabetesYes                            0.0007357323
#> ulcerYes                               0.4180133887
```

```
#> strokeYes                              .
#> emphysemaYes                          -0.0509549802
#> copdYes                                0.1002374105
#> high.cholesterolYes                    0.1021030868
#> coronary.heart.diseaseYes              0.0220558291
#> anginaYes                              0.0849595261
#> heart.attackYes                        0.2476656673
#> heart.diseaseYes                       .
#> arthritisYes                           0.9746692568
#> crohns.diseaseYes                      .
#> place.routine.careDoctor's office     -0.0583977619
#> place.routine.careHospital/Clinic      .
#> place.routine.careOther place          .
#> trouble.asleepYes                      0.2030296869
#> obeseYes                               0.3879895531
#> current.smokerYes                      0.3374178965
#> heavy.drinkerYes                      -0.1268971235
#> hospitalization1-2 days                0.1192556336
#> hospitalization3-5 days                .
#> hospitalization6+ days                 1.0866087297
#> better.health.statusYes               -0.1013785074
#> physical.activityModerate             -0.7523741651
#> physical.activityHigh                 -0.6865233686
fit.lasso[[2]]$beta
#> 67 x 1 sparse Matrix of class "dgCMatrix"
#>                                                s0
#> sexMale                                .
#> hhsize                                 0.018493189
#> bornOther place                        .
#> maritalMarried/with partner            .
#> maritalDivorced/separated             -0.001903136
#> maritalWidowed                         .
#> regionMidwest                         -0.077656662
#> regionSouth                            0.066061919
#> regionWest                             0.198988965
#> raceBlack                             -0.313705357
#> raceHispanic                          -0.160881871
#> raceOthers                             .
#> educationHigh school/GED              -0.073948767
#> educationSome college                  .
```

```
#> educationBachelors degree or higher  -0.092601336
#> employment.statusEmployed non-hourly  .
#> employment.statusWorked previously    0.662301617
#> employment.statusNever worked         0.990458783
#> poverty.status100-200% FPL            .
#> poverty.status200-400% FPL           -0.253983648
#> poverty.status400%+ FPL              -0.485846823
#> veteranYes                           -0.095649252
#> insuranceMedicaid/Medicare            .
#> insurancePrivately Insured           -0.005530756
#> insuranceOther                        0.249278523
#> sex.orientationOther                  .
#> worried.moneyYes                      0.301627606
#> good.neighborhoodYes                 -0.554793692
#> psy.symptomYes                        0.975043141
#> visit.EDOne                           0.058872693
#> visit.ED2-3                           0.255785667
#> visit.ED4+                            0.422172434
#> surgeryOne                            0.002831717
#> surgeryTwo                            0.099101540
#> surgery3+                             .
#> dr.visit6-12 months                  -0.105087667
#> dr.visit1-5 years                    -0.381233762
#> dr.visit>5 years/never               -0.459656177
#> cancerYes                             0.281041121
#> asthmaYes                             0.427654297
#> htnYes                                0.219625784
#> liver.diseaseYes                      0.580991873
#> diabetesYes                           .
#> ulcerYes                              0.348542693
#> strokeYes                             0.070369016
#> emphysemaYes                          .
#> copdYes                               .
#> high.cholesterolYes                   0.150862244
#> coronary.heart.diseaseYes             0.009536678
#> anginaYes                             .
#> heart.attackYes                       0.405053759
#> heart.diseaseYes                      .
#> arthritisYes                          0.954063592
#> crohns.diseaseYes                     .
```

```
#> place.routine.careDoctor's office    -0.045764606
#> place.routine.careHospital/Clinic     .
#> place.routine.careOther place         0.207253975
#> trouble.asleepYes                     0.212898902
#> obeseYes                              0.389340100
#> current.smokerYes                     0.332982403
#> heavy.drinkerYes                     -0.135194457
#> hospitalization1-2 days               .
#> hospitalization3-5 days               .
#> hospitalization6+ days                1.018420971
#> better.health.statusYes              -0.001173805
#> physical.activityModerate            -0.703703292
#> physical.activityHigh                -0.677811398
fit.lasso[[3]]$beta
#> 67 x 1 sparse Matrix of class "dgCMatrix"
#>                                            s0
#> sexMale                              -0.03555390
#> hhsize                                0.01276707
#> bornOther place                       .
#> maritalMarried/with partner          -0.00226132
#> maritalDivorced/separated             .
#> maritalWidowed                        .
#> regionMidwest                         .
#> regionSouth                           .
#> regionWest                            0.25300309
#> raceBlack                            -0.21629021
#> raceHispanic                          .
#> raceOthers                            0.07458935
#> educationHigh school/GED             -0.07527970
#> educationSome college                 .
#> educationBachelors degree or higher  -0.08524865
#> employment.statusEmployed non-hourly  .
#> employment.statusWorked previously    0.66693373
#> employment.statusNever worked         0.96274685
#> poverty.status100-200% FPL            .
#> poverty.status200-400% FPL           -0.11543598
#> poverty.status400%+ FPL              -0.31422327
#> veteranYes                           -0.08849244
#> insuranceMedicaid/Medicare            .
#> insurancePrivately Insured           -0.14856615
```

```
#> insuranceOther                          0.14368311
#> sex.orientationOther                    .
#> worried.moneyYes                        0.28107756
#> good.neighborhoodYes                   -0.37214169
#> psy.symptomYes                          1.02129791
#> visit.EDOne                             0.11427725
#> visit.ED2-3                             0.23654896
#> visit.ED4+                              0.53011900
#> surgeryOne                              0.06030144
#> surgeryTwo                              .
#> surgery3+                              -0.28355643
#> dr.visit6-12 months                    -0.04219568
#> dr.visit1-5 years                      -0.83131719
#> dr.visit>5 years/never                 -0.48832578
#> cancerYes                               0.18366379
#> asthmaYes                               0.13556093
#> htnYes                                  0.12622357
#> liver.diseaseYes                        0.62123990
#> diabetesYes                             .
#> ulcerYes                                0.39650846
#> strokeYes                               .
#> emphysemaYes                            .
#> copdYes                                 0.16563326
#> high.cholesterolYes                     0.10541633
#> coronary.heart.diseaseYes               0.08229669
#> anginaYes                               .
#> heart.attackYes                         0.37154172
#> heart.diseaseYes                        .
#> arthritisYes                            0.91902311
#> crohns.diseaseYes                      -0.01478240
#> place.routine.careDoctor's office      -0.04968533
#> place.routine.careHospital/Clinic       .
#> place.routine.careOther place           0.20094580
#> trouble.asleepYes                       0.07725267
#> obeseYes                                0.40542559
#> current.smokerYes                       0.27207489
#> heavy.drinkerYes                       -0.06932537
#> hospitalization1-2 days                -0.13525647
#> hospitalization3-5 days                 .
#> hospitalization6+ days                  1.02580763
```

```
#> better.health.statusYes                .
#> physical.activityModerate            -0.74686507
#> physical.activityHigh                -0.90554222
fit.lasso[[4]]$beta
#> 67 x 1 sparse Matrix of class "dgCMatrix"
#>                                                s0
#> sexMale                                 .
#> hhsize                                  .
#> bornOther place                         .
#> maritalMarried/with partner             .
#> maritalDivorced/separated               .
#> maritalWidowed                          .
#> regionMidwest                           .
#> regionSouth                             .
#> regionWest                              0.151732262
#> raceBlack                              -0.063747960
#> raceHispanic                            .
#> raceOthers                              .
#> educationHigh school/GED                .
#> educationSome college                   .
#> educationBachelors degree or higher     .
#> employment.statusEmployed non-hourly    .
#> employment.statusWorked previously      0.521879242
#> employment.statusNever worked           0.671265401
#> poverty.status100-200% FPL              .
#> poverty.status200-400% FPL             -0.014851029
#> poverty.status400%+ FPL                -0.229122884
#> veteranYes                              .
#> insuranceMedicaid/Medicare              .
#> insurancePrivately Insured             -0.092128877
#> insuranceOther                          .
#> sex.orientationOther                    0.150490732
#> worried.moneyYes                        0.282333603
#> good.neighborhoodYes                   -0.104236603
#> psy.symptomYes                          1.132555465
#> visit.EDOne                             0.009230339
#> visit.ED2-3                             0.288663967
#> visit.ED4+                              0.673934923
#> surgeryOne                              .
#> surgeryTwo                              .
```

```
#> surgery3+                                .
#> dr.visit6-12 months                     -0.018831608
#> dr.visit1-5 years                       -0.496368171
#> dr.visit>5 years/never                  -0.415165313
#> cancerYes                                .
#> asthmaYes                                0.240957776
#> htnYes                                   0.125845727
#> liver.diseaseYes                         0.672282863
#> diabetesYes                              .
#> ulcerYes                                 0.352645788
#> strokeYes                                .
#> emphysemaYes                             .
#> copdYes                                  0.054275559
#> high.cholesterolYes                      0.057735457
#> coronary.heart.diseaseYes                0.070550439
#> anginaYes                                .
#> heart.attackYes                          0.234807175
#> heart.diseaseYes                         .
#> arthritisYes                             1.043934915
#> crohns.diseaseYes                        0.086038829
#> place.routine.careDoctor's office        .
#> place.routine.careHospital/Clinic        .
#> place.routine.careOther place            .
#> trouble.asleepYes                        0.177916442
#> obeseYes                                 0.363088024
#> current.smokerYes                        0.339985811
#> heavy.drinkerYes                         .
#> hospitalization1-2 days                  .
#> hospitalization3-5 days                  0.073321609
#> hospitalization6+ days                   1.066617646
#> better.health.statusYes                  .
#> physical.activityModerate               -0.699175264
#> physical.activityHigh                   -0.642594150
fit.lasso[[5]]$beta
#> 67 x 1 sparse Matrix of class "dgCMatrix"
#>                                                s0
#> sexMale                                  .
#> hhsize                                   0.01585559
#> bornOther place                          .
#> maritalMarried/with partner              .
```

```
#> maritalDivorced/separated            -0.04982466
#> maritalWidowed                        0.03837620
#> regionMidwest                        -0.25040909
#> regionSouth                           .
#> regionWest                            0.08288425
#> raceBlack                            -0.07020324
#> raceHispanic                          .
#> raceOthers                            .
#> educationHigh school/GED              .
#> educationSome college                 .
#> educationBachelors degree or higher   .
#> employment.statusEmployed non-hourly  .
#> employment.statusWorked previously    0.64497666
#> employment.statusNever worked         0.83560645
#> poverty.status100-200% FPL            .
#> poverty.status200-400% FPL           -0.03113577
#> poverty.status400%+ FPL              -0.22313059
#> veteranYes                           -0.20269803
#> insuranceMedicaid/Medicare            .
#> insurancePrivately Insured           -0.14170986
#> insuranceOther                        0.19604683
#> sex.orientationOther                  .
#> worried.moneyYes                      0.31572735
#> good.neighborhoodYes                 -0.31850186
#> psy.symptomYes                        1.15194136
#> visit.EDOne                           0.05961452
#> visit.ED2-3                           0.34349175
#> visit.ED4+                            0.25637410
#> surgeryOne                            .
#> surgeryTwo                            .
#> surgery3+                             .
#> dr.visit6-12 months                   .
#> dr.visit1-5 years                    -0.32790023
#> dr.visit>5 years/never                .
#> cancerYes                             0.21372688
#> asthmaYes                             0.21206722
#> htnYes                                0.13025900
#> liver.diseaseYes                      0.44892643
#> diabetesYes                           0.05119678
#> ulcerYes                              0.27325347
```

```
#> strokeYes                              .
#> emphysemaYes                           .
#> copdYes                                0.04954730
#> high.cholesterolYes                    0.14824572
#> coronary.heart.diseaseYes              .
#> anginaYes                              .
#> heart.attackYes                        0.27924438
#> heart.diseaseYes                       .
#> arthritisYes                           1.00382986
#> crohns.diseaseYes                      .
#> place.routine.careDoctor's office     -0.01943640
#> place.routine.careHospital/Clinic      .
#> place.routine.careOther place          0.07273335
#> trouble.asleepYes                      0.14684831
#> obeseYes                               0.38029871
#> current.smokerYes                      0.14111139
#> heavy.drinkerYes                       .
#> hospitalization1-2 days               -0.02310396
#> hospitalization3-5 days                .
#> hospitalization6+ days                 1.08731840
#> better.health.statusYes                .
#> physical.activityModerate             -0.76639891
#> physical.activityHigh                 -0.38546251
```

```
# AUCs from different folds
auc.lasso
#> [1] 0.8396619 0.8129805 0.8465035 0.7896878 0.8342721

# Calibration slope from different folds
cal.slope.lasso
#> [1] 1.1287166 0.9985467 1.1224240 0.8934633 1.0131154

# Brier score from different folds
brier.lasso
#> [1] 0.08524781 0.08476661 0.08666820 0.09071329 0.08911735
```

Now we will average out the model performance measures:

```r
# Average AUC
mean(auc.lasso)
#> [1] 0.8246212

# Average calibration slope
mean(cal.slope.lasso)
#> [1] 1.031253

# Average Brier score
mean(brier.lasso)
#> [1] 0.08730265
```

Although the authors used multiple imputation, our AUC from the LASSO model with complete case data analysis is not that different. Note: the authors reported the AUC values in Table 2.

Random forest for surveys

Now, we will fit the random forest model for predicting binary HICP with the listed predictors. Here are the steps for fitting the model with 5-fold CV:

- For fold 1, folds 2-5 is the training set and fold 1 is the test set
- Fit random forest model on the training set to find the value of the hyperparameters (number of trees, number of predictors to split at in each node, and minimal node size to split at) that gives minimum prediction error. Incorporate sampling weights in the model to account for survey design.
- Grid-search with out-of-sample error approach is widely used in the literature. In this approach, we create a data frame from all combinations of the hyperparameters and check which combination gives the lowest out-of-sample error.
- Fit the random forest model on the training with the selected hyperparameters from the previous step. Incorporate sampling weights in the model to account for survey design.

- Calculate predictive performance (e.g., AUC) on the test set
- Repeat the analysis for all folds.

Folds

```
k <- 5
table(nfolds)
#> nfolds
#>    1    2    3    4    5
#> 1451 1457 1496 1468 1408
```

Formula

```
Formula
#> HICP ~ sex + hhsize + born + marital + region + race + education +
#>     employment.status + poverty.status + veteran + insurance +
#>     sex.orientation + worried.money + good.neighborhood + psy.symptom +
#>     visit.ED + surgery + dr.visit + cancer + asthma + htn + liver.disease +
#>     diabetes + ulcer + stroke + emphysema + copd + high.cholesterol +
#>     coronary.heart.disease + angina + heart.attack + heart.disease +
#>     arthritis + crohns.disease + place.routine.care + trouble.asleep +
#>     obese + current.smoker + heavy.drinker + hospitalization +
#>     better.health.status + physical.activity
```

5-fold CV random forest

```
fit.rf <- list(NULL)
auc.rf <- NULL
cal.slope.rf <- brier.rf <- NULL
for (fold in 1:k) {
  # Training data
  dat.train <- dat.complete[nfolds != fold, ]

  # Test data
```

```r
dat.test <- dat.complete[nfolds == fold, ]

# Tuning the hyperparameters
## Grid with 1000 models - huge time consuming
#grid.search <- expand.grid(mtry = 1:10, node.size = 1:10,
#                           num.trees = seq(50,500,50),
#                            OOB_RMSE = 0)

## Grid with 36 models as an exercise
grid.search <- expand.grid(
  mtry = 5:7,
  node.size = 1:3,
  num.trees = seq(200,500,100),
  OOB_RMSE = 0)

## Model with grids
for(ii in 1:nrow(grid.search)) {
  # Model on training set with grid
  fit.rf.tune <- ranger(
    formula = Formula,
    data = dat.train,
    num.trees = grid.search$num.trees[ii],
    mtry = grid.search$mtry[ii],
    min.node.size = grid.search$node.size[ii],
    importance = 'impurity',
    case.weights = dat.train$wgt)

  # Add Out-of-bag (OOB) error to grid
  grid.search$OOB_RMSE[ii] <-
    sqrt(fit.rf.tune$prediction.error)
}
# Position of the tuned hyperparameters
position <- which.min(grid.search$OOB_RMSE)

# Fit the model on the training set with tuned hyperparameters
fit.rf[[fold]] <- ranger(
  formula = Formula,
  data = dat.train,
  case.weights = dat.train$wgt,
  probability = T,
```

```r
    num.trees = grid.search$num.trees[position],
    min.node.size = grid.search$node.size[position],
    mtry = grid.search$mtry[position],
    importance = 'impurity')

  # Prediction on the test set
  dat.test$pred.rf <- predict(
    fit.rf[[fold]],
    data = dat.test)$predictions[,2]

  # AUC on the test set with sampling weights
  auc.rf[fold] <- WeightedAUC(
    WeightedROC(dat.test$pred.rf,
                dat.test$HICP,
                weight = dat.test$wgt))

  # Weighted calibration slope
  dat.test$pred.rf[dat.test$pred.rf == 0] <- 0.00001
  mod.cal <- glm(HICP ~ Logit(dat.test$pred.rf),
                 data = dat.test,
                 family = binomial,
                 weights = wgt)
  cal.slope.rf[fold] <- summary(mod.cal)$coef[2,1]

  # Weighted Brier Score
  brier.rf[fold] <- mean(brierscore(
    HICP ~ dat.test$pred.rf,
    data = dat.test,
    wt = dat.test$wgt))
}
```

Model performance

Let's check how prediction worked.

```r
# Fitted random forest models
fit.rf[[1]]
#> Ranger result
#>
```

```
#> Call:
#>  ranger(formula = Formula, data = dat.train, case.weights = dat.train$wgt,      probability
#>
#> Type:                             Probability estimation
#> Number of trees:                  500
#> Sample size:                      5829
#> Number of independent variables:  42
#> Mtry:                             5
#> Target node size:                 3
#> Variable importance mode:         impurity
#> Splitrule:                        gini
#> OOB prediction error (Brier s.):  0.0899798
fit.rf[[2]]
#> Ranger result
#>
#> Call:
#>  ranger(formula = Formula, data = dat.train, case.weights = dat.train$wgt,      probability
#>
#> Type:                             Probability estimation
#> Number of trees:                  400
#> Sample size:                      5823
#> Number of independent variables:  42
#> Mtry:                             5
#> Target node size:                 2
#> Variable importance mode:         impurity
#> Splitrule:                        gini
#> OOB prediction error (Brier s.):  0.0899217
fit.rf[[3]]
#> Ranger result
#>
#> Call:
#>  ranger(formula = Formula, data = dat.train, case.weights = dat.train$wgt,      probability
#>
#> Type:                             Probability estimation
#> Number of trees:                  500
#> Sample size:                      5784
#> Number of independent variables:  42
#> Mtry:                             5
#> Target node size:                 3
#> Variable importance mode:         impurity
```

```
#> Splitrule:                          gini
#> OOB prediction error (Brier s.):  0.08972587
fit.rf[[4]]
#> Ranger result
#>
#> Call:
#>  ranger(formula = Formula, data = dat.train, case.weights = dat.train$wgt,      probability
#>
#> Type:                             Probability estimation
#> Number of trees:                  400
#> Sample size:                      5812
#> Number of independent variables:  42
#> Mtry:                             5
#> Target node size:                 3
#> Variable importance mode:         impurity
#> Splitrule:                        gini
#> OOB prediction error (Brier s.):  0.08934626
fit.rf[[5]]
#> Ranger result
#>
#> Call:
#>  ranger(formula = Formula, data = dat.train, case.weights = dat.train$wgt,      probability
#>
#> Type:                             Probability estimation
#> Number of trees:                  400
#> Sample size:                      5872
#> Number of independent variables:  42
#> Mtry:                             5
#> Target node size:                 3
#> Variable importance mode:         impurity
#> Splitrule:                        gini
#> OOB prediction error (Brier s.):  0.08929408
```

```
# AUCs from different folds
auc.rf
#> [1] 0.8393204 0.7905715 0.8244523 0.7811141 0.8301950

# Calibration slope from different folds
cal.slope.rf
#> [1] 1.2842866 0.9933553 1.1486583 0.9134864 1.2263184
```

```r

# Brier score from different folds
brier.rf
#> [1] 0.08745016 0.08881079 0.08913696 0.09163393 0.08895764
```

Now we will average out the model performance measures:

```r
# Average AUC
mean(auc.rf)
#> [1] 0.8131307

# Average calibration slope
mean(cal.slope.rf)
#> [1] 1.113221

# Average Brier score
mean(brier.rf)
#> [1] 0.0891979
```

This AUC from random forest is approximately the same as obtained from the LASSO model.

Variable importance

One nice feature of random forest is that we can rank the variables and generate a variable importance plot.

```r
# Fold 1
ggplot(
  enframe(fit.rf[[1]]$variable.importance,
          name = "variable",
          value = "importance"),
  aes(x = reorder(variable, importance),
      y = importance, fill = importance)) +
  geom_bar(stat = "identity",
           position = "dodge") +
  coord_flip() +
  ylab("Variable Importance") +
  xlab("") +
```

```
  ggtitle("") +
  guides(fill = "none") +
  scale_fill_gradient(low = "grey",
                      high = "grey10") +
  theme_bw()
```

```

# Fold 5
ggplot(
  enframe(fit.rf[[5]]$variable.importance,
          name = "variable",
          value = "importance"),
  aes(x = reorder(variable, importance),
      y = importance, fill = importance)) +
  geom_bar(stat = "identity",
           position = "dodge") +
  coord_flip() +
  ylab("Variable Importance") +
  xlab("") +
  ggtitle("") +
  guides(fill = "none") +
  scale_fill_gradient(low = "grey",
                      high = "grey10") +
  theme_bw()
```

hospitalization
psy.symptom
region
poverty.status
education
visit.ED
arthritis
marital
hhsize
insurance
physical.activity
place.routine.care
surgery
worried.money
race
obese
trouble.asleep
employment.status
high.cholesterol
cancer
diabetes
hiv
sex
heart.disease
asthma
ulcer
better.health.status
good.neighborhood
copd
coronary.heart.disease
heart.attack
stroke
veteran
current.smoker
drvisit
born
angina
liver.disease
emphysema
crohns.disease
sex.orientation
heavy.drinker
0
20
40
60
Variable Importance

References

Quiz (L)

> ⚠ Warning
>
> We are currently in the beta testing phase for our quiz page updates, so please be aware that some responses may not be accurate. Thank you for your patience.
> For those interested in downloading the quiz questions, you'll find a section below dedicated to downloading.

Live Quiz

You can click on the live quiz options to either see hints of why your response may be incorrect (response marked in red), or whether your choice of response is correct (response marked in green):

Download Quiz

1. **Downloading the File:**
   - Navigate to the link: See here.
   - Right-click on the link and select "Save link as…" from the dropdown menu.
   - Alternatively click here for downloading the quizzes.
   - Choose a destination folder on your computer where you'd like to save the file (e.g., Desktop). **Remember this location**, as you'll need to navigate to it later.
2. **Setting Up RStudio:**

- If you don't have RStudio installed, see the download link in here.
- Launch RStudio after installing.

3. **Installing Necessary R Packages:**

   - Before running the R Markdown file, ensure you have the required packages. In RStudio's console (located at the bottom left by default), enter the following commands to install them:

```r
install.packages("learnr")
install.packages("xfun")
```

4. **Opening and Running the File:**

   - In RStudio, go to `File` > `Open File...`.
   - Navigate to the folder where you saved the `Rmd` file and select it to open.
   - Once the file is open in RStudio, you'll see a "Run Document" button (green) at the top of the script editor. Click on it to run the R Markdown Quiz file.

R functions (L)

The list of new R functions introduced in this *Machine learning* lab component are below:

Function_name	Package_name	Use
fancyRpartPlot	rattle	To plot an rpart object
fviz_nbclust	factoextra	To visualize the optimal number of clusters
kmeans	base/stats	To conduct K-Means cluster analysis
lowess	base/stats	To perform scatter plot smoothing aka lowess smoothing
rpart	rpart	To fit a classification tree (CART)
terms	base/stats	To extarct terms objects
varImp	caret	To calculate the variable importance measure

Part X

ML in causal inference

Background

This chapter provides a detailed exploration of Targeted Maximum Likelihood Estimation (TMLE) in causal inference. The first tutorial motivates the use of TMLE by highlighting its advantages over traditional methods, focusing on the limitations of these approaches and introducing the RHC dataset. The second tutorial delves into the SuperLearner method for ensemble modeling, discussing the importance of algorithm diversity, cross-validation, and adaptable libraries. The third tutorial offers a comprehensive guide to applying TMLE for binary outcomes, emphasizing diverse SuperLearner libraries, effective sample sizes, and candidate learner selection. The fourth tutorial extends TMLE to continuous outcomes, covering transformations, interpretation, and comparisons with default TMLE libraries and traditional regression. These tutorials should equip readers with a robust understanding of TMLE and its practical applications in causal inference and epidemiology.

> **!** Important
>
> **Datasets**:
> All of the datasets used in this tutorial can be accessed from this GitHub repository folder

Overview of tutorials

Motivation for learning and using TMLE

This tutorial discusses the motivation for using the TMLE method in causal inference. It highlights the limitations of traditional methods such as propensity score approaches and direct application of machine learning in terms of assumptions and statistical inference. TMLE is presented as a doubly robust method that incorporates machine learning while allowing straightforward statistical inference. The tutorial reintroduces RHC data for demonstration.

In the preceding chapters, we have developed a solid understanding of predictive modeling, data splitting/cross-validation, propensity scores and various machine learning algorithms. These insights have prepared us for the advanced causal inference techniques we will encounter in this chapter. As we explore TMLE and SuperLearner, we will draw upon our knowledge of propensity score and machine learning to unlock the potential of their use in building powerful causal inference tools.

Understanding SuperLearner

This tutorial focuses on using the SuperLearner method for ensemble modeling. SuperLearner is a type 2 ensemble method that combines various predictive algorithms to create a robust predictive model. It employs cross-validation to determine the best-weighted combination of algorithms, based on specified predictive performance metrics. The tutorial provides guidelines for selecting a diverse set of algorithms, considering computational feasibility, and adapting the library of algorithms based on sample characteristics. Examples of different types of learners that can be included in the SuperLearner library are provided, ranging from parametric to highly data-adaptive and non-linear models. The tutorial also mentions the default libraries for estimating outcomes and propensity scores in the context of TMLE.

Dealing with binary outcomes within TMLE framework

This tutorial is a comprehensive guide to applying TMLE for binary outcomes. It covers the entire TMLE process, from constructing initial outcome and exposure models to targeted adjustment via propensity scores and treatment effect estimation. It emphasizes the importance of specifying a diverse SuperLearner library for both exposure and outcome models, determining effective sample sizes, and selecting candidate learners. The tutorial demonstrates TMLE application using the `tmle` package and includes a thorough comparison with default SuperLearner libraries and traditional regression, presenting estimates, confidence intervals, and a comparative table of results.

Dealing with continuous outcomes within TMLE framework

This tutorial offers comprehensive guidance on implementing TMLE for continuous outcomes, including transformations and result interpretation. It introduces the application of TMLE

for continuous outcomes, emphasizing the process of constructing a SuperLearner and key considerations such as effective sample size and learner selection. Notably, it highlights the essential transformation of continuous outcome variables to a standardized range (0 to 1) using min-max normalization before applying TMLE with a Gaussian family. The tutorial demonstrates the post-TMLE rescaling of treatment effect estimates and confidence intervals to the original scale and offers a comparative analysis with default TMLE libraries and traditional regression.

Comparing results

In this comprehensive tutorial, various statistical methods were applied to investigate the association between RHC and death using the RHC dataset. These methods included logistic regression, propensity score matching and weighting with both logistic regression and Super Learner, as well as TMLE. The results consistently indicated that participants with RHC use had higher odds of death compared to those without RHC use, with odds ratios ranging similarly across different modeling approaches, but also showing a trend when machine learners are incorporated.

> Tip
>
> For those who prefer a video walkthrough of the additional resources, feel free to watch the video below.

> Warning
>
> **Bug Report**:
> Fill out this form to report any issues with the tutorial.

References

Concepts (C)

ML in causal inference

In comparative effectiveness studies, researchers typically use propensity score methods. However, propensity score methods have known limitations in real-world scenarios, when the true data generating mechanism is unknown. Targeted maximum likelihood estimation (TMLE) is an alternative estimation method with a number of desirable statistical properties. It is a doubly robust method, enabling the integration of machine learning approaches within the framework. Despite the fact that this method has been shown to perform better in terms of statistical properties (e.g., variance estimation) than propensity score methods in a variety of scenarios, it is not widely used in medical research as the implementation details of this approach are generally not well understood. In this section, we will explain this method in details.

Reading list

Key reference: (E. Karim and Frank 2021)

Optional reading: (Frank and Karim 2023)

Video Lessons

Machine learning

The workshop was first developed for R/Medicine Virtual Conference https://r-medicine.org/ 2021, August 24th. What is included in this Video Lesson/workshop:

- Chapter 1 RHC data description 4:22
- Chapter 2 G-computation 23:13
- Chapter 3 G-computation using ML 45:02
- Chapter 4 IPTW 1:18:50
- Chapter 5 IPTW using ML 1:30:11
- Chapter 6 TMLE 1:36:41
- Chapter 7 Pre-packaged software 1:58:05
- Chapter 8 Final Words 2:14:36

The timestamps are also included in the YouTube video description.

Links

The original workshop materials are available here.

References

Motivation

- When using methods like propensity score approaches, we are making assumptions about the model specification. For example, we must specify any interaction terms.
- With machine learning methods, these assumptions can be relaxed somewhat, as some machine learning methods allow automatic detection of data structures such as interactions.
- However, machine learning was created for prediction modeling, not with causal inference in mind. Statistical inference such as calculating standard errors and confidence intervals is not as straightforward since the estimator given by machine learning methods does not follow a known statistical distribution. By contrast, the estimators resulting from a standard regression using maximum likelihood estimation will follow an approximately normal distribution, where it is easy to calculate standard errors and confidence intervals.
- **Targeted maximum likelihood estimation (TMLE)** is a causal inference method that can incorporate machine learning in a way that still allows straightforward statistical inference based on theoretical development grounded in semi-parametric theory.
  - TMLE is a **doubly robust** method. This means it uses both the exposure (AKA propensity score) model *and* the outcome model. As long as *one* of these models is correctly specified, TMLE will give a **consistent estimator**, meaning it gets closer and closer to the true value as the sample size increases.
  - Since TMLE uses both the exposure and the outcome model, machine learning can be used in each

of these intermediary modeling steps while allowing straightforward statistical inference.

- It has been shown that TMLE outperform singly robust methods with machine learning, such as IPTW.

Revisiting RHC Data

> **Note**
>
> This tutorial uses the same data as some of the previous tutorials, including working with a predictive question, machine learning with a continuous outocome, and machine learning with a binary outcome.

```r
ObsData <- readRDS(file =
                     "Data/machinelearningCausal/rhcAnalyticTest.RDS")
baselinevars <- names(dplyr::select(ObsData,
                         !c(RHC.use,Length.of.Stay,Death)))
head(ObsData)
#>   Disease.category          Cancer Cardiovascular Congestive.HF Dementia
#> 1            Other Localized (Yes)              0             0        0
#> 2             MOSF            None              1             1        0
#> 3             MOSF Localized (Yes)              0             0        0
#> 4              ARF            None              0             0        0
#> 5             MOSF            None              0             0        0
#> 6            Other            None              0             1        0
#>   Psychiatric Pulmonary Renal Hepatic GI.Bleed Tumor Immunosupperssion
#> 1           0         1     0       0        0     1                 0
#> 2           0         0     0       0        0     0                 1
#> 3           0         0     0       0        0     1                 1
#> 4           0         0     0       0        0     0                 1
#> 5           0         0     0       0        0     0                 0
#> 6           0         1     0       0        0     0                 0
#>   Transfer.hx MI       age    sex       edu DASIndex APACHE.score
#> 1           0  0   [70,80)   Male 12.000000 23.50000           46
#> 2           1  0   [70,80) Female 12.000000 14.75195           50
#> 3           0  0 [-Inf,50) Female 14.069916 18.13672           82
#> 4           0  0   [70,80) Female  9.000000 22.92969           48
```

```
#> 5           0  0   [60,70)   Male  9.945259 21.05078           72
#> 6           0  0 [80, Inf) Female  8.000000 17.50000           38
#>   Glasgow.Coma.Score blood.pressure         WBC Heart.rate Respiratory.rate
#> 1                  0             41 22.09765620        124               10
#> 2                  0             63 28.89843750        137               38
#> 3                  0             57  0.04999542        130               40
#> 4                  0             55 23.29687500         58               26
#> 5                 41             65 29.69921880        125               27
#> 6                  0            115 18.00000000        134               36
#>   Temperature PaO2vs.FIO2  Albumin Hematocrit Bilirubin Creatinine Sodium
#> 1    38.69531     68.0000 3.500000   58.00000 1.0097656  1.1999512    145
#> 2    38.89844    218.3125 2.599609   32.50000 0.6999512  0.5999756    137
#> 3    36.39844    275.5000 3.500000   21.09766 1.0097656  2.5996094    146
#> 4    35.79688    156.6562 3.500000   26.29688 0.3999634  1.6999512    117
#> 5    34.79688    478.0000 3.500000   24.00000 1.0097656  3.5996094    126
#> 6    39.19531    184.1875 3.099609   30.50000 1.0097656  1.3999023    138
#>   Potassium PaCo2       PH   Weight DNR.status  Medical.insurance
#> 1  4.000000    40 7.359375 64.69995         No           Medicare
#> 2  3.299805    34 7.329102 45.69998         No Private & Medicare
#> 3  2.899902    16 7.359375  0.00000         No            Private
#> 4  5.799805    30 7.459961 54.59998         No Private & Medicare
#> 5  5.799805    17 7.229492 78.39996        Yes           Medicare
#> 6  5.399414    68 7.299805 54.89999         No           Medicare
#>   Respiratory.Diag Cardiovascular.Diag Neurological.Diag Gastrointestinal.Diag
#> 1              Yes                 Yes               No                    No
#> 2               No                  No               No                    No
#> 3               No                 Yes               No                    No
#> 4              Yes                  No               No                    No
#> 5               No                 Yes               No                    No
#> 6              Yes                  No               No                    No
#>   Renal.Diag Metabolic.Diag Hematologic.Diag Sepsis.Diag Trauma.Diag
#> 1         No             No               No          No          No
#> 2         No             No               No         Yes          No
#> 3         No             No               No          No          No
#> 4         No             No               No          No          No
#> 5         No             No               No          No          No
#> 6         No             No               No          No          No
#>   Orthopedic.Diag  race     income Length.of.Stay Death RHC.use
#> 1              No white Under $11k              9     0       0
#> 2              No white Under $11k             45     1       1
```

```
#> 3              No white   $25-$50k            60     0       1
#> 4              No white   $11-$25k            37     1       0
#> 5              No white Under $11k             2     1       1
#> 6              No white Under $11k             7     0       0
```

Table 1

Only for some demographic and comorbidity variables; matches with Table 1 in Connors et al. (1996).

```
tab0 <- CreateTableOne(vars = c("age", "sex", "race",
                                "Disease.category", "Cancer"),
                       data = ObsData,
                       strata = "RHC.use",
                       test = FALSE)
print(tab0, showAllLevels = FALSE)
#>                        Stratified by RHC.use
#>                         0            1
#>   n                     3551         2184
#>   age (%)
#>      [-Inf,50)           884 (24.9)   540 (24.7)
#>      [50,60)             546 (15.4)   371 (17.0)
#>      [60,70)             812 (22.9)   577 (26.4)
#>      [70,80)             809 (22.8)   529 (24.2)
#>      [80, Inf)           500 (14.1)   167 ( 7.6)
#>   sex = Female (%)      1637 (46.1)   906 (41.5)
#>   race (%)
#>      white              2753 (77.5)  1707 (78.2)
#>      black               585 (16.5)   335 (15.3)
#>      other               213 ( 6.0)   142 ( 6.5)
#>   Disease.category (%)
#>      ARF                1581 (44.5)   909 (41.6)
#>      CHF                 247 ( 7.0)   209 ( 9.6)
#>      Other               955 (26.9)   208 ( 9.5)
#>      MOSF                768 (21.6)   858 (39.3)
#>   Cancer (%)
#>      None               2652 (74.7)  1727 (79.1)
#>      Localized (Yes)     638 (18.0)   334 (15.3)
#>      Metastatic          261 ( 7.4)   123 ( 5.6)
```

References

SuperLearner

Choosing Learners

SuperLearner is a *type 2* ensemble method, meaning it combines many methods of different types into one predictive model. SuperLearner uses cross-validation to find the best weighted combination of algorithms based on the predictive performance measure specified (default in the `SuperLearner` package is non-negative least squares based on the Lawson-Hanson algorithm (Mullen and Stokkum 2023), but measures such as AUC can also be used). To run SuperLearner, the user needs to specify a *library* consisting of all the different methods SuperLearner should incorporate in the final model, as well as the number of cross-validation folds.

SuperLearner will perform as well as possible *given the library of algorithms considered*. A very recent paper by Phillips et al. (2023) provides some concrete guidelines for the determination of the number of cross-validation folds necessary and the selection of algorithms to include. Overall, we want to make sure the set of algorithms provided is:

See previous chapter for other types of ensemble learning methods.

- **Diverse**: Having a rich library of algorithms allows the SuperLearner to adapt to a range of underlying data structures. Diverse libraries include:
  - Parametric learners such as generalized linear models (GLMs)
  - Highly data-adaptive learners
  - Multiple variants of the same learner with different parameter specifications
- **Computationally feasible**: Lots of machine learning algorithms take a long time to run. Having multiple computationally intensive algorithms in your library will cause

the SuperLearner as a whole to take much too long to run.

> **Note**
>
> Some of the more specific guidelines depend on our effective sample size. For binary outcomes, this can be calculated as:
>
> $$n_{eff} = min(n, 5 * (n * min(\bar{p}, 1 - \bar{p})))$$
>
> where $\bar{p}$: prevalence of the outcome.
> For continuous outcomes, the effective sample size is the same as the sample size ($n_{eff} = n$).

We also want to consider the characteristics of our particular sample.

- **If there are continuous covariates**: We should include learners that do not force relationships to be linear/monotonic. For example, we could include regression splines, support vector machines, and tree-based methods like regression trees.
- **If we have high-dimensional data (a large number of covariates e.g. more than $n_{eff}/20$)**: We should include some learners that fall under the class of *screeners*. These are learners which incorporate dimension reduction such as LASSO and random forests.
- **If the sample size is very large** (i.e. $n_{eff} > 500$): We should include as many learners as is computationally feasible.
- **If the sample size is small** (i.e. $n_{eff} \leq 500$): We should include fewer learners (e.g. up to $n_{eff}/5$), and include less flexible learners.

Some examples of learners that could be included are given in the table below (Polley 2021):

Type of learner	Examples
Parametric	• `SL.mean`: simple mean • `SL.glm`: generalized linear models • `SL.lm`: ordinary least squares • `SL.speedglm`: fast version of glm • `SL.speedlm`: fast version of lm • `SL.gam`: generalized additive methods • `SL.step`: choose model based on AIC (backwards or forwards or both)

Highly data-adaptive	• `SL.glmnet`: penalized regression using elastic net (ridge regression and Lasso)
	• Kernel-based methods
	  – `SL.kernelKnn`: k-nearest neighbours
	  – `SL.ksvm`: kernel-based support vector machine
	• `SL.xgboost`: extreme gradient boosting
	• `SL.gbm`: gradient-boosted machines
	• `SL.nnet`: neural networks

Type of learner	Examples

Allowing non-linear/monotonic relationships	• `SL.earth`: multivariate adaptive regression splines
	• Tree-based methods
	  – `SL.randomForest`: random forests
	  – `tmle.SL.dbarts2`: bayesian additive regression trees
	  – `SL.cforest`: random forests using conditional inference trees
	  – `SL.ranger`: fast implementation of random forest suited for high dimensional data
	• `SL.svm`: support vector machines

Screeners	• `screen.corP`: retain covariates with correlation with outcome p-value <0.1 • `screen.corRank`: retain top j covariates with highest correlation with outcome • `screen.glmnet`: Lasso • `screen.randomForest`: random forests • `screen.SIS`: retain covariates based on distance correlation

There is also a useful tool implemented in the `SuperLearner` library which allows us to easily see a list of all available learners.

```
SuperLearner::listWrappers()
#> All prediction algorithm wrappers in SuperLearner:
```

```
#>  [1] "SL.bartMachine"      "SL.bayesglm"         "SL.biglasso"
#>  [4] "SL.caret"            "SL.caret.rpart"      "SL.cforest"
#>  [7] "SL.earth"            "SL.extraTrees"       "SL.gam"
#> [10] "SL.gbm"              "SL.glm"              "SL.glm.interaction"
#> [13] "SL.glmnet"           "SL.ipredbagg"        "SL.kernelKnn"
#> [16] "SL.knn"              "SL.ksvm"             "SL.lda"
#> [19] "SL.leekasso"         "SL.lm"               "SL.loess"
#> [22] "SL.logreg"           "SL.mean"             "SL.nnet"
#> [25] "SL.nnls"             "SL.polymars"         "SL.qda"
#> [28] "SL.randomForest"     "SL.ranger"           "SL.ridge"
#> [31] "SL.rpart"            "SL.rpartPrune"       "SL.speedglm"
#> [34] "SL.speedlm"          "SL.step"             "SL.step.forward"
#> [37] "SL.step.interaction" "SL.stepAIC"          "SL.svm"
#> [40] "SL.template"         "SL.xgboost"
#>
#> All screening algorithm wrappers in SuperLearner:
#> [1] "All"
#> [1] "screen.corP"           "screen.corRank"        "screen.glmnet"
#> [4] "screen.randomForest"   "screen.SIS"            "screen.template"
#> [7] "screen.ttest"          "write.screen.template"
```

SuperLearner in TMLE

- The default SuperLearner library for estimating the outcome includes (S. Gruber, Van Der Laan, and Kennedy 2020)
  - `SL.glm`: generalized linear models (GLMs)
  - `SL.glmnet`: least absolute shrinkage and selection operator (LASSO)
  - `tmle.SL.dbarts2`: modeling and prediction using Bayesian Additive Regression Trees (BART)
- The default library for estimating the propensity scores includes
  - `SL.glm`: generalized linear models (GLMs)
  - `tmle.SL.dbarts.k.5`: SL wrappers for modeling and prediction using BART
  - `SL.gam`: generalized additive models (GAMs)

- It is certainly possible to use different set of learners
  - More methods can be added by
    - specifying lists of models in the *Q.SL.library* (for the outcome model) and *g.SL.library* (for the propensity score model)

References

Binary Outcomes

For this example we will be looking at the binary outcome variable `death`.

```r
# Data
load(file = "Data/machinelearningCausal/cl2.RData")

# Table 1
tab1 <- CreateTableOne(vars = c("Death"),
                       data = ObsData,
                       strata = "RHC.use",
                       test = FALSE)
print(tab1, showAllLevels = FALSE, )
#>                     Stratified by RHC.use
#>                      0           1
#>   n                  3551        2184
#>   Death (mean (SD)) 0.63 (0.48) 0.68 (0.47)
```

TMLE

TMLE works by first constructing an initial outcome and extracting a crude estimate of the treatment effect. Then, TMLE aims to refine the initial estimate in the direction of the true value of the parameter of interest through use of the exposure model.

To label the treatment effect given by TMLE as causal, the same conditional exchangeability, positivity, and consistency assumptions must be met as for other modeling strategies (see introduction to propensity score). TMLE also assumes that at least one of the exposure or outcome model is correctly specified. If this does not hold, TMLE does not necessarily produce a consistent estimator.

The basic steps are:

1. Construct initial outcome model & get crude estimate
2. Construct exposure model and use propensity scores to update the initial outcome model through a targeted adjustment
3. Extract treatment effect estimate
4. Estimate confidence interval based on a closed-form formula

Luque-Fernandez et al. (2018) discussed the implementation of TMLE, and providing a detailed step-by-step guide, primarily focusing on a binary outcome.

The tmle package implements TMLE for both binary and continuous outcomes, and uses the SuperLearner to construct the exposure and outcome models.

The *tmle* method takes a number of parameters, including:

```
#> Warning: package 'knitr' was built under R version 4.2.3
```

Term	Description
Y	Outcome vector
A	Exposure vector
W	Matrix that includes vectors of all covariates
family	Distribution
V	Cross-validation folds for exposure and outcome modeling
Q.SL.library	Set of machine learning methods to use for SuperLearner for outcome modeling
g.SL.library	Set of machine learning methods to use for SuperLearner for exposure modeling

Constructing SuperLearner

We will need to specify two SuperLearners, one for the exposure and one for the outcome model. We will need to consider the characteristics of our sample in order to decide the number of cross-validation folds and construct a diverse and computationally feasible library of algorithms.

Number of folds

First, we need to define the number of cross-validation folds to use for each model. This depends on our effective sample size (Phillips et al. 2023).

Our effective sample size for the outcome model is:

```
n <- nrow(ObsData)
p <- nrow(ObsData[ObsData$Death == 1,])/n
n_eff <- min(n, 5*(n*min(p, 1-p)))
n_eff
#> [1] 5735
```

Our effective sample size for the exposure model is:

```
p_exp <- nrow(ObsData[ObsData$RHC.use == 1,])/n
n_eff_exp <- min(n, 5*(n*min(p, 1-p)))
n_eff_exp
#> [1] 5735
```

For both models, the effective sample size is the same as our sample size, $n = 5,735$.

Since $5,000 \leq n_{eff} \leq 10,000$, we should use 5 or more cross-validation folds according to Phillips et al. (2023). For the sake of computational feasibility, we will use **5 folds** in this example.

Candidate learners

The second step is to define the library of learners we will feed in to SuperLearner as potential options for each model (exposure and outcome). In this example, some of our covariates are continuous variables, such as temperature and blood pressure, so we need to include learners that allow non-linear/monotonic relationships.

Since n is large (> 5000), we should include as many learners as is computationally feasible in our libraries.

Furthermore, we have 50 covariates:

```r
length(c(baselinevars, "Length.of.Stay"))
#> [1] 50
```

5735/20 = 286.75, and 50 < 286.75, so we do not have high-dimensional data and including screeners is optional (Phillips et al. 2023).

Since the requirements for the exposure and outcome models are the same in this example, we will use the same SuperLearner library for both. Overall for this example we need to make sure to include:

- Parametric learners
- Highly data-adaptive learners
- Multiple variants of the same learner with different parameter specifications
- Learners that allow non-linear/monotonic relationships

For this example, we will include the following learners:

- Parametric
  - `SL.mean`: only mean
  - `SL.glm`: generalized linear model
- Highly data-adaptive
  - `SL.glmnet`: penalized regression such as lasso
  - `SL.xgboost`: extreme gradient boosting
- Allowing non-linear/monotonic relationships
  - `SL.randomForest`: random forest
  - `tmle.SL.dbarts2`: bayesian additive regression tree
  - `SL.svm`: support vector machine

```r
# Construct the SuperLearner library
SL.library <- c("SL.mean",
                "SL.glm",
                "SL.glmnet",
                "SL.xgboost",
                "SL.randomForest",
                "tmle.SL.dbarts2",
                "SL.svm")
```

TMLE with SuperLearner

To run TMLE, we need to install the tmle package and load it on the R environment.

```r
#install.packages(c('tmle', 'xgboost'))
require(tmle)
require(xgboost)
```

We also need to create a data frame containing only the covariates:

```r
ObsData.noYA <- dplyr::select(ObsData,
                              !c(Death, RHC.use))
ObsData$Death <- as.numeric(ObsData$Death)
```

Then we can run TMLE using the `tmle` method from the `tmle` package:

```r
set.seed(1444)

tmle.fit <- tmle::tmle(Y = ObsData$Death,
                   A = ObsData$RHC.use,
                   W = ObsData.noYA,
                   family = "binomial",
                   V.Q = 5,
                   V.g = 5,
                   Q.SL.library = SL.library,
                   g.SL.library = SL.library)

```

```r
tmle.est.bin <- tmle.fit$estimates$OR$psi
tmle.ci.bin <- tmle.fit$estimates$OR$CI
```

ATE for binary outcome using user-specified library: 1.28 and 95% CI is 1.1992981, 1.3761869

These results show those who received RHC had odds of death that were 1.28 times as high as the odds of death in those who did not receive RHC.

Understanding defaults

We can compare the results using our specified SuperLearner library to the results we would get when using the `tmle` package's default SuperLearner libraries. To do this we simply do not specify libraries for the `Q.SL.library` and `g.SL.library` arguments.

```r
# small test library
# with only glm just
# for sake of making this work
SL.library.test <- c("SL.glm")
```

```r
set.seed(1444)

tmle.fit.def <- tmle::tmle(Y = ObsData$Death,
                           A = ObsData$RHC.use,
                           W = ObsData.noYA,
                           family = "binomial",
                           V.Q = 5,
                           V.g = 5)
# Q.SL.library = SL.library.test,  ## removed this line
# g.SL.library = SL.library.test)  ## removed this line

tmle.est.bin.def <- tmle.fit.def$estimates$OR$psi
tmle.ci.bin.def <- tmle.fit.def$estimates$OR$CI
```

ATE for binary outcome using default library: 1.32 with 95% CI 1.1466162, 1.5219877.

These ATE when using the default SuperLearner library (1.31) is very close to the ATE when using our user-specified SuperLearner library (1.29). However, the confidence interval from TMLE using the default SuperLearner library (1.17, 1.46) is slightly wider than the confidence interval from TMLE using our user-specified SuperLearner library (1.20, 1.39).

Comparison of results

We can also compare these results to those from a basic regression and from the literature.

```r
# adjust the exposure variable
# (primary interest) + covariates
baselineVars.Death <- c(baselinevars, "Length.of.Stay")
out.formula.bin <- as.formula(
  paste("Death~ RHC.use +",
        paste(baselineVars.Death,
              collapse = "+")))
fit1.bin <- glm(out.formula.bin, data = ObsData, family="binomial")
```

Connors et al. (1996) conducted a propensity score matching analysis. Table 4 showed that, after propensity score pair (1-to-1) matching, the odds of in-hospital mortality were 39% higher in those who received RHC (OR: 1.39 (1.15, 1.67)).

method.list	Estimate	2.5 %	97.5 %
Adjusted Regression	0.07	0.04	0.09
TMLE (user-specified SL library)	1.28	1.20	1.38
TMLE (default SL library)	1.32	1.15	1.52
Connors et al. (1996) paper	1.39	1.15	1.67

References

Continuous Outcomes

We will now go through an example of using TMLE for a continuous outcome. The setup for SuperLearner in this case is similar to that for binary outcomes, so rather than going through the SuperLearner steps again, we will instead focus on the additional steps that are necessary for running the `tmle` method on continuous outcomes.

> **Note**
>
> Only outcome variable (Length of stay); slightly different than Table 2 in Connors et al. (1996) (means were 20.5 vs. 25.7; and medians were 16 vs. 17).

Frank and Karim (2023) extensively discussed the implementation of TMLE for continuous outcomes, providing a detailed step-by-step guide using the openly accessible RHC dataset. In this tutorial, we will revisit the same example with additional explanations.

```r
tab1 <- CreateTableOne(vars = c("Length.of.Stay"),
                       data = ObsData,
                       strata = "RHC.use",
                       test = FALSE)
print(tab1, showAllLevels = FALSE, )
#>                              Stratified by RHC.use
#>                               0              1
#>   n                            3551           2184
#>   Length.of.Stay (mean (SD)) 19.53 (23.59) 24.86 (28.90)
```

```r
median(ObsData$Length.of.Stay[ObsData$RHC.use==0])
#> [1] 12
median(ObsData$Length.of.Stay[ObsData$RHC.use==1])
#> [1] 16
```

Constructing SuperLearner

Just as we did for a binary outcome, we will need to specify two SuperLearners, one for the exposure and one for the outcome model.

The effective sample size for a continuous outcome is just $n_{eff} = n = 5,735$. We calculated the effective sample size for the exposure model earlier, which also turned out to be $n_{eff} = n = 5,735$. So once again we will use *5 folds* because $5,000 \leq n_{eff} \leq 10,000$ (Phillips et al. 2023).

Similarly to our example with the binary outcome, the key considerations for the library of learners are:

- We have some continuous covariates, and should therefore include learners that allow non-linear/monotonic relationships.
- We have a large n, so should include as many learners as is computationally feasible.
- We have 49 covariates and 5,735 observations, so we do not have high-dimensional data and including screeners is optional.

Again the requirements for the exposure and outcome models are the same and we can use the same library for both models. Note that even though one model will have a binary dependent variable, and one will have a continuous dependent variable, most of the available learners automatically adapt to binary and continuous dependent variables.

For this example, we will use the same SuperLearner library as for the binary outcome example.

```r
# Construct the SuperLearner library
SL.library <- c("SL.mean",
                "SL.glm",
                "SL.glmnet",
                "SL.xgboost",
                "SL.randomForest",
                "tmle.SL.dbarts2",
                "SL.svm")
```

Dealing with continuous outcomes

For this example, we will be examining the length of stay in hospital outcome.

The key difference between running TMLE on a continuous outcome in comparison to running it with a binary outcome, is that we must **transform** the outcome to fall within the range of 0 to 1, so that the modeled outcomes fall within the range of the outcome's true distribution (Susan Gruber and Laan 2010).

To transform the outcome, we can use min-max normalization:

$$Y_{transformed} = \frac{Y - Y_{min}}{Y_{max} - Y_{min}}$$

```
set.seed(1444)
# transform the outcome to fall within the range [0,1]
min.Y <- min(ObsData$Length.of.Stay)
max.Y <- max(ObsData$Length.of.Stay)
ObsData$Length.of.Stay_transf <-
  (ObsData$Length.of.Stay-min.Y)/
  (max.Y-min.Y)
```

Once we have transformed the outcome to fall within the range of 0 to 1, we can run TMLE as before, using the `tmle` method in the `tmle` package:

```
# create data frame containing only covariates
ObsData.noYA <- dplyr::select(ObsData,
                              !c(Length.of.Stay_transf,
                                 Length.of.Stay,
                                 RHC.use))
```

```
set.seed(1444)

# run tmle
tmle.fit.cont <- tmle::tmle(Y = ObsData$Length.of.Stay_transf,
                        A = ObsData$RHC.use,
```

```
                          W = ObsData.noYA,
                          family = "gaussian",
                          V.Q = 5,
                          V.g = 5,
                          Q.SL.library = SL.library,
                          g.SL.library = SL.library)
```

Once the `tmle` method has run, we still have one step to complete to get our final estimate. At this point, we must transform the average treatment effect generated by the `tmle` method ($\widehat{ATE}$) back to the outcome's original scale:

$$\widehat{ATE}_{rescaled} = (Y_{max} - Y_{min}) * \widehat{ATE}$$

```
# transform back the ATE estimate
tmle.est.cont <- (max.Y-min.Y)*
  tmle.fit.cont$estimates$ATE$psi
tmle.est.cont
#> [1] 2.939622
```

We also have to transform the confidence interval back to the original scale:

```
tmle.ci.cont <- (max.Y-min.Y)*
  tmle.fit.cont$estimates$ATE$CI
```

ATE for continuous outcome: 2.9396218, and 95 % CI is 1.959698, 3.9195455.

The results indicate that if all participants had received RHC, the average length of stay in hospital would be 2.95 (1.99, 3.91) days longer than if no participants had received RHC.

Understanding defaults

Transform outcome:

```r
set.seed(1444)
# transform the outcome to fall within the range [0,1]
min.Y <- min(ObsData$Length.of.Stay)
max.Y <- max(ObsData$Length.of.Stay)
ObsData$Length.of.Stay_transf <-
  (ObsData$Length.of.Stay-min.Y)/
  (max.Y-min.Y)
```

Run TMLE, using the `tmle` package's default SuperLearner library:

```r
# create data frame containing only covariates
ObsData.noYA <- dplyr::select(ObsData,
                              !c(Length.of.Stay_transf,
                                 Length.of.Stay,
                                 RHC.use))
```

```r
set.seed(1444)

# run tmle
tmle.fit.cont.def <- tmle::tmle(
  Y = ObsData$Length.of.Stay_transf,
  A = ObsData$RHC.use,
  W = ObsData.noYA,
  family = "gaussian",
  V.Q = 5,
  V.g = 5)
# Q.SL.library = SL.library.test,
## removed this line
# g.SL.library = SL.library.test)
## removed this line
```

Transform the average treatment effect generated by the `tmle` method ($\widehat{ATE}$) back to the outcome's original scale:

$$\widehat{ATE}_{rescaled} = (Y_{max} - Y_{min}) * \widehat{ATE}$$

```r
# transform back the ATE estimate
tmle.est.cont.def <- (max.Y-min.Y)*
  tmle.fit.cont.def$estimates$ATE$psi
tmle.est.cont.def
#> [1] 3.352891
```

Transform the confidence interval back to the original scale:

```r
tmle.ci.cont.def <- (max.Y-min.Y)*
  tmle.fit.cont.def$estimates$ATE$CI
```

ATE for continuous outcome using default library: 3.0362984, and 95% CI 1.2686301, 4.8039667.

The estimate using the default SuperLearner library (2.18) is similar to the estimate we got when using our user-specified SuperLearner library (2.95). However, the confidence interval using the default SuperLearner library (1.25, 4.37) was much wider than that using our user-specified SuperLearner library (1.99, 3.91).

Comparison of results

Adjusted regression:

```r
# adjust the exposure variable
# (primary interest) + covariates
baselineVars.LoS <- c(baselinevars, "Death")
out.formula.cont <- as.formula(
  paste("Length.of.Stay~ RHC.use +",
        paste(baselineVars.LoS,
              collapse = "+")))
fit1.cont <- lm(out.formula.cont, data = ObsData)
publish(fit1.cont, digits=1)$regressionTable[2,]
```

Connors et al. (1996) conducted a propensity score matching analysis. Table 5 showed that, after propensity score pair (1-to-1) matching, means of length of stay (Y), when stratified by RHC (A) were not significantly different ($p = 0.14$).

method.list	Estimate	2.5 %	97.5 %
Adjusted Regression	3.04	1.51	4.58
TMLE (user-specified SL library)	5.34	3.90	6.78
TMLE (default SL library)	3.35	1.43	4.86
Keele and Small (2021) paper	2.01	0.60	3.41

Differences in results can likely be attributed to the use of different SuperLearner libraries, the use of different combinations of variables used, or random sampling associated with the cross-validation used in the SuperLearner algorithm.

References

Comparing results

In this tutorial, we will use the RHC dataset in exploring the relationship between RHC (yes/no) and death (yes/no). We will use the following approaches:

- logistic regression
- propensity score matching and weighting with logistic regression
- propensity score matching and weighting with Super Learner
- TMLE

Load packages

We load several R packages required for fitting the models.

```
# Load required packages
library(tableone)
library(Publish)
library(randomForest)
library(tmle)
library(xgboost)
library(kableExtra)
library(SuperLearner)
library(dbarts)
library(MatchIt)
library(cobalt)
library(survey)
library(knitr)
```

Load dataset

```r
load(file = "Data/machinelearningCausal/cl2.RData")
ls()
#> [1] "baselinevars"    "has_annotations" "ObsData"         "tab0"
```

```r
# Data
dat <- ObsData
head(dat)
#>   Disease.category          Cancer Cardiovascular Congestive.HF Dementia
#> 1            Other Localized (Yes)              0             0        0
#> 2             MOSF            None              1             1        0
#> 3             MOSF Localized (Yes)              0             0        0
#> 4              ARF            None              0             0        0
#> 5             MOSF            None              0             0        0
#> 6            Other            None              0             1        0
#>   Psychiatric Pulmonary Renal Hepatic GI.Bleed Tumor Immunosupperssion
#> 1           0         1     0       0        0     1                 0
#> 2           0         0     0       0        0     0                 1
#> 3           0         0     0       0        0     1                 1
#> 4           0         0     0       0        0     0                 1
#> 5           0         0     0       0        0     0                 0
#> 6           0         1     0       0        0     0                 0
#>   Transfer.hx MI       age    sex       edu DASIndex APACHE.score
#> 1           0  0   [70,80)   Male 12.000000 23.50000           46
#> 2           1  0   [70,80) Female 12.000000 14.75195           50
#> 3           0  0 [-Inf,50) Female 14.069916 18.13672           82
#> 4           0  0   [70,80) Female  9.000000 22.92969           48
#> 5           0  0   [60,70)   Male  9.945259 21.05078           72
#> 6           0  0 [80, Inf) Female  8.000000 17.50000           38
#>   Glasgow.Coma.Score blood.pressure         WBC Heart.rate Respiratory.rate
#> 1                  0             41 22.09765620       124               10
#> 2                  0             63 28.89843750       137               38
#> 3                  0             57  0.04999542       130               40
#> 4                  0             55 23.29687500        58               26
#> 5                 41             65 29.69921880       125               27
#> 6                  0            115 18.00000000       134               36
#>   Temperature PaO2vs.FIO2  Albumin Hematocrit Bilirubin Creatinine Sodium
#> 1    38.69531     68.0000 3.500000   58.00000 1.0097656  1.1999512    145
#> 2    38.89844    218.3125 2.599609   32.50000 0.6999512  0.5999756    137
```

```
#> 3    36.39844    275.5000 3.500000   21.09766 1.0097656  2.5996094    146
#> 4    35.79688    156.6562 3.500000   26.29688 0.3999634  1.6999512    117
#> 5    34.79688    478.0000 3.500000   24.00000 1.0097656  3.5996094    126
#> 6    39.19531    184.1875 3.099609   30.50000 1.0097656  1.3999023    138
#>   Potassium PaCo2       PH   Weight DNR.status  Medical.insurance
#> 1  4.000000    40 7.359375 64.69995         No           Medicare
#> 2  3.299805    34 7.329102 45.69998         No Private & Medicare
#> 3  2.899902    16 7.359375  0.00000         No            Private
#> 4  5.799805    30 7.459961 54.59998         No Private & Medicare
#> 5  5.799805    17 7.229492 78.39996        Yes           Medicare
#> 6  5.399414    68 7.299805 54.89999         No           Medicare
#>   Respiratory.Diag Cardiovascular.Diag Neurological.Diag Gastrointestinal.Diag
#> 1              Yes                 Yes                No                    No
#> 2               No                  No                No                    No
#> 3               No                 Yes                No                    No
#> 4              Yes                  No                No                    No
#> 5               No                 Yes                No                    No
#> 6              Yes                  No                No                    No
#>   Renal.Diag Metabolic.Diag Hematologic.Diag Sepsis.Diag Trauma.Diag
#> 1         No             No               No          No          No
#> 2         No             No               No         Yes          No
#> 3         No             No               No          No          No
#> 4         No             No               No          No          No
#> 5         No             No               No          No          No
#> 6         No             No               No          No          No
#>   Orthopedic.Diag  race     income Length.of.Stay Death RHC.use
#> 1              No white Under $11k              9     0       0
#> 2              No white Under $11k             45     1       1
#> 3              No white   $25-$50k             60     0       1
#> 4              No white   $11-$25k             37     1       0
#> 5              No white Under $11k              2     1       1
#> 6              No white Under $11k              7     0       0

# Data dimension
dim(dat)
#> [1] 5735   52
```

Confounder list in the `baselinevars` vector is

Confounders
Disease.category
Cancer
Cardiovascular
Congestive.HF
Dementia
Psychiatric
Pulmonary
Renal
Hepatic
GI.Bleed
Tumor
Immunosupperssion
Transfer.hx
MI
age
sex
edu
DASIndex
APACHE.score
Glasgow.Coma.Score
blood.pressure
WBC
Heart.rate
Respiratory.rate
Temperature
PaO2vs.FIO2
Albumin
Hematocrit
Bilirubin
Creatinine
Sodium
Potassium
PaCo2
PH
Weight
DNR.status
Medical.insurance
Respiratory.Diag
Cardiovascular.Diag

Confounders
Neurological.Diag
Gastrointestinal.Diag
Renal.Diag
Metabolic.Diag
Hematologic.Diag
Sepsis.Diag
Trauma.Diag
Orthopedic.Diag
race
income

Logistic regression

Let us define the regression formula and fit logistic regression, adjusting for baseline confounders.

```
# Formula
Formula <- formula(paste("Death ~ RHC.use + ",
                         paste(baselinevars,
                               collapse = " + ")))

# Logistic regression
fit.glm <- glm(Formula,
               data = dat,
               family = binomial)
```

We can use `flextable` package to view `fit.glm`, the regression output:

Variable	Odds_Ratio	CI_Lower	CI_Upper
RHC.use	1.42	1.23	1.65

As we can see, the odds of death was 42% higher among those participants with RHC use than those with no RHC use.

Propensity score matching with logistic regression

Now we will use propensity score matching, where propensity score will be estimated using logistic regression. The first step is to define the propensity score formula and estimate the propensity scores.

Step 1

```r
# Propensity score model define
ps.formula <- as.formula(paste0("RHC.use ~ ",
                                paste(baselinevars,
                                      collapse = "+")))

# Propensity score model fitting
fit.ps <- glm(ps.formula,
              data = dat,
              family = binomial)

# Propensity scores
dat$ps <- predict(fit.ps,
                  type = "response")
summary(dat$ps)
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> 0.002478 0.161446 0.358300 0.380819 0.574319 0.968425
```

Step 2

The second step is to match exposed (i.e., RHC users) to unexposed (i.e., RHC non-users) based on the propensity scores. We will use nearest neighborhood and caliper approach.

```r
# Caliper
caliper <- 0.2*sd(log(dat$ps/(1-dat$ps)))

# 1:1 matching
set.seed(123)
match.obj <- matchit(ps.formula,
```

```
                        data = dat,
                        distance = dat$ps,
                        method = "nearest",
                        ratio = 1,
                        caliper = caliper)
```

```r
# See how many matched
match.obj
#> A matchit object
#>  - method: 1:1 nearest neighbor matching without replacement
#>  - distance: User-defined [caliper]
#>  - caliper: <distance> (0.068)
#>  - number of obs.: 5735 (original), 3478 (matched)
#>  - target estimand: ATT
#>  - covariates: too many to name

# Extract matched data
matched.data <- match.data(match.obj)

# Overlap checking
bal.plot(match.obj,
         var.name="distance",
         which="both",
         type = "density",
         colors = c("red","blue"))
```

Step 3

The third step is to check the balancing on the matched data. We will compare the similarity of baseline characteristics between RHC users and non-users in the propensity score matched sample. Let's consider SMD >0.1 as imbalanced.

```
# Balance checking in terms of SMD - using love plot
love.plot(match.obj,
          binary = "std",
          grid = TRUE,
          thresholds = c(m = .1),
          colors = c("red","blue"))
```

```

# Balance checking in terms of SMD - using tableone
tab1b <- CreateTableOne(vars = baselinevars, strata = "RHC.use",
                        data = matched.data, includeNA = T,
                        addOverall = F, test = F)
#print(tab1b, showAllLevels = T, catDigits = 2, smd = T)
```

`ExtractSmd(tab1b)` shows

Variables	1 vs 2
Disease.category	0.06
Cancer	0.03
Cardiovascular	0.03
Congestive.HF	0.00
Dementia	0.01
Psychiatric	0.04
Pulmonary	0.04
Renal	0.03
Hepatic	0.01
GI.Bleed	0.00
Tumor	0.01
Immunosupperssion	0.01
Transfer.hx	0.03
MI	0.01
age	0.04
sex	0.02
edu	0.03
DASIndex	0.03
APACHE.score	0.08
Glasgow.Coma.Score	0.01
blood.pressure	0.07
WBC	0.01
Heart.rate	0.00
Respiratory.rate	0.04
Temperature	0.02
PaO2vs.FIO2	0.06

Variables	1 vs 2
Albumin	0.03
Hematocrit	0.03
Bilirubin	0.03
Creatinine	0.03
Sodium	0.02
Potassium	0.01
PaCo2	0.04
PH	0.02
Weight	0.02
DNR.status	0.00
Medical.insurance	0.06
Respiratory.Diag	0.06
Cardiovascular.Diag	0.05
Neurological.Diag	0.04
Gastrointestinal.Diag	0.01
Renal.Diag	0.02
Metabolic.Diag	0.01
Hematologic.Diag	0.01
Sepsis.Diag	0.03
Trauma.Diag	0.02
Orthopedic.Diag	0.05
race	0.02
income	0.06

After propensity score matching, all the confounders are balanced in terms of SMD. Now, we will fit the outcome model on the matched data.

Step 4

```r
fit.psm <- glm(Death ~ RHC.use,
               data = matched.data,
               family = binomial)
```

Summary of `fit.psm`:

Variable	Odds_Ratio	CI_Lower	CI_Upper
RHC.use	1.29	1.12	1.48

In the propensity score matched data, the odds of death was 29% higher among those participants with RHC use than those with no RHC use.

Propensity score weighting with logistic regression

Now we will use the propensity score weighting approach where propensity scores are estimated using logistic regression.

Step 1

Step 1 is the same as we did it for the propensity score matching.

Step 2

For the second step, we will calculate the stabilized inverse probability weight.

```r
dat$ipw <- with(dat, ifelse(RHC.use==1, mean(RHC.use)/ps,
                            mean(1-RHC.use)/(1-ps)))
summary(dat$ipw)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.3932  0.6571  0.7781  0.9953  1.0624 24.1854
```

Step 3

Now, we will check the balance in terms of SMD.

```r
# Design with inverse probability weights
w.design <- svydesign(id = ~1, weights = ~ipw, data = dat, nest = F)

# Balance checking in terms of SMD
tab1e <- svyCreateTableOne(vars = baselinevars, strata = "RHC.use",
                           data = w.design, includeNA = T,
                           addOverall = F, test = F)
#print(tab1e, showAllLevels = T, catDigits = 2, smd = T)
```

`ExtractSmd(tab1e)` shows

Variables	1 vs 2
Disease.category	0.02
Cancer	0.01
Cardiovascular	0.01
Congestive.HF	0.00
Dementia	0.05
Psychiatric	0.02
Pulmonary	0.02
Renal	0.01
Hepatic	0.00
GI.Bleed	0.02
Tumor	0.00
Immunosupperssion	0.01
Transfer.hx	0.01
MI	0.00
age	0.05
sex	0.03

Variables	1 vs 2
edu	0.00
DASIndex	0.04
APACHE.score	0.01
Glasgow.Coma.Score	0.00
blood.pressure	0.01
WBC	0.04
Heart.rate	0.02
Respiratory.rate	0.00
Temperature	0.01
PaO2vs.FIO2	0.00
Albumin	0.03
Hematocrit	0.02
Bilirubin	0.01
Creatinine	0.01
Sodium	0.01
Potassium	0.03
PaCo2	0.02
PH	0.01
Weight	0.02
DNR.status	0.04
Medical.insurance	0.03
Respiratory.Diag	0.01
Cardiovascular.Diag	0.01
Neurological.Diag	0.01
Gastrointestinal.Diag	0.01
Renal.Diag	0.01

Variables	1 vs 2
Metabolic.Diag	0.00
Hematologic.Diag	0.00
Sepsis.Diag	0.01
Trauma.Diag	0.01
Orthopedic.Diag	0.01
race	0.02
income	0.02

All confounders are balanced (SMD < 0.1).

Step 4

```
fit.ipw <- svyglm(Death ~ RHC.use,
                  design = w.design,
                  family = binomial)
```

Summary of `fit.ipw`:

Variable	Odds_Ratio	CI_Lower	CI_Upper
RHC.use	1.30	1.11	1.53

In the propensity score weighted data, the odds of death was 30% higher among those participants with RHC use than those with no RHC use.

Propensity score matching with super learner

Now we will use the propensity score matching, where we will be using super learner to calculate the propensity scores. We use logistic, LASSO, and XGBoost as the candidate learners.

Step 1

```r
set.seed(123)
ps.fit <- CV.SuperLearner(
  Y = dat$RHC.use,
  X = dat[,baselinevars],
  family = "binomial",
  SL.library = c("SL.glm", "SL.glmnet", "SL.xgboost"),
  verbose = FALSE,
  V = 5,
  method = "method.NNLS")
```

```r
# Propensity scores for all learners
predictions <- cbind(ps.fit$SL.predict, ps.fit$library.predict)
head(predictions)
#>                SL.glm_All SL.glmnet_All SL.xgboost_All
#> [1,] 0.3632635 0.39789364    0.39569431     0.32144672
#> [2,] 0.5870492 0.64983590    0.61497078     0.54465985
#> [3,] 0.5740246 0.66329951    0.65396650     0.46847290
#> [4,] 0.2172501 0.33044244    0.34478278     0.02363573
#> [5,] 0.6347749 0.45972157    0.43779434     0.86645132
#> [6,] 0.0272532 0.03420477    0.03629319     0.01730520

# Propensity scores for super learner
dat$ps.sl <- predictions[,1]
summary(dat$ps.sl)
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> 0.001671 0.147762 0.337664 0.376282 0.587237 0.975584
```

Step 2

The same as before, we will match exposed to unexposed based on their propensity scores.

```r
# Caliper
caliper <- 0.2*sd(log(dat$ps.sl/(1-dat$ps.sl)))

# 1:1 matching
```

```r
set.seed(123)
match.obj2 <- matchit(ps.formula,
                      data = dat,
                      distance = dat$ps.sl,
                      method = "nearest",
                      ratio = 1,
                      caliper = caliper)
```

```r
# See how many matched
match.obj2
#> A matchit object
#>  - method: 1:1 nearest neighbor matching without replacement
#>  - distance: User-defined [caliper]
#>  - caliper: <distance> (0.075)
#>  - number of obs.: 5735 (original), 3430 (matched)
#>  - target estimand: ATT
#>  - covariates: too many to name

# Extract matched data
matched.data.sl <- match.data(match.obj2)

# Overlap checking
bal.plot(match.obj2,
         var.name="distance",
         which="both",
         type = "density",
         colors = c("red","blue"))
```

Step 3

Now we will check the balancing on the matched data.

```
# Balance checking in terms of SMD - using love plot
love.plot(match.obj2,
          binary = "std",
          grid = TRUE,
          thresholds = c(m = .1),
          colors = c("red","blue"))
```

```r
# Balance checking in terms of SMD - using tableone
tab1c <- CreateTableOne(vars = baselinevars, strata = "RHC.use",
                        data = matched.data.sl, includeNA = T,
                        addOverall = T, test = F)
#print(tab1c, showAllLevels = T, catDigits = 2, smd = T)
```

`ExtractSmd(tab1c)` shows

Variables	1 vs 2
Disease.category	0.08
Cancer	0.05
Cardiovascular	0.01
Congestive.HF	0.02
Dementia	0.00
Psychiatric	0.02
Pulmonary	0.01
Renal	0.01
Hepatic	0.02
GI.Bleed	0.02
Tumor	0.05
Immunosupperssion	0.01
Transfer.hx	0.06
MI	0.03
age	0.06
sex	0.04
edu	0.03
DASIndex	0.01
APACHE.score	0.08
Glasgow.Coma.Score	0.01

Variables	1 vs 2
blood.pressure	0.05
WBC	0.00
Heart.rate	0.05
Respiratory.rate	0.03
Temperature	0.04
PaO2vs.FIO2	0.09
Albumin	0.05
Hematocrit	0.03
Bilirubin	0.00
Creatinine	0.05
Sodium	0.01
Potassium	0.01
PaCo2	0.05
PH	0.06
Weight	0.06
DNR.status	0.04
Medical.insurance	0.06
Respiratory.Diag	0.08
Cardiovascular.Diag	0.05
Neurological.Diag	0.04
Gastrointestinal.Diag	0.02
Renal.Diag	0.00
Metabolic.Diag	0.02
Hematologic.Diag	0.02
Sepsis.Diag	0.04
Trauma.Diag	0.06

Variables	1 vs 2
Orthopedic.Diag	0.06
race	0.02
income	0.04

Again, all confounders are balanced in terms of SMD (all SMDs < 0.1). Next, we will fit the outcome model.

Step 4

```
fit.psm.sl <- glm(Death ~ RHC.use,
                  data = matched.data.sl,
                  family = binomial)
```

Summary of `fit.psm.sl`:

Variable	Odds_Ratio	CI_Lower	CI_Upper
RHC.use	1.26	1.09	1.45

The interpretation is the same as before. In the propensity score matched data, the odds of death was 26% higher among those participants with RHC use than those without RHC use.

Propensity score weighting with super learner

Step 1

Step 1 is the same as we did it for the propensity score matching.

Step 2

For the second step, we will calculate the stabilized inverse probability weight.

```
dat$ipw.sl <- with(dat, ifelse(RHC.use==1, mean(RHC.use)/ps.sl,
                               mean(1-RHC.use)/(1-ps.sl)))
summary(dat$ipw.sl)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.3903  0.6496  0.7647  1.0191  1.0468 41.4490
```

Step 3

Next, we will check the balance in terms of SMD.

```
# Design with inverse probability weights
w.design.sl <- svydesign(id = ~1, weights = ~ipw.sl,
                         data = dat, nest = F)

# Balance checking in terms of SMD
tab1k <- svyCreateTableOne(vars = baselinevars, strata = "RHC.use",
                           data = w.design.sl, includeNA = T,
                           addOverall = T, test = F)
#print(tab1k, showAllLevels = T, catDigits = 2, smd = T)
```

`ExtractSmd(tab1k)` shows

Variables	1 vs 2
Disease.category	0.01
Cancer	0.03
Cardiovascular	0.02
Congestive.HF	0.01
Dementia	0.05
Psychiatric	0.03
Pulmonary	0.01

Variables	1 vs 2
Renal	0.01
Hepatic	0.00
GI.Bleed	0.02
Tumor	0.00
Immunosupperssion	0.01
Transfer.hx	0.00
MI	0.01
age	0.07
sex	0.03
edu	0.00
DASIndex	0.04
APACHE.score	0.01
Glasgow.Coma.Score	0.02
blood.pressure	0.05
WBC	0.06
Heart.rate	0.00
Respiratory.rate	0.01
Temperature	0.00
PaO2vs.FIO2	0.03
Albumin	0.00
Hematocrit	0.02
Bilirubin	0.01
Creatinine	0.00
Sodium	0.00
Potassium	0.02
PaCo2	0.05

Variables	1 vs 2
PH	0.02
Weight	0.01
DNR.status	0.05
Medical.insurance	0.03
Respiratory.Diag	0.02
Cardiovascular.Diag	0.01
Neurological.Diag	0.02
Gastrointestinal.Diag	0.00
Renal.Diag	0.01
Metabolic.Diag	0.01
Hematologic.Diag	0.01
Sepsis.Diag	0.00
Trauma.Diag	0.05
Orthopedic.Diag	0.02
race	0.05
income	0.05

All confounders are balanced (SMD < 0.1).

Step 4

```
fit.ipw.sl <- svyglm(Death ~ RHC.use,
                     design = w.design.sl,
                     family = binomial)
```

Summary of `fit.ipw.sl`:

Variable	Odds_Ratio	CI_Lower	CI_Upper
RHC.use	1.26	1.04	1.52

In the propensity score weighted data, the odds of death was 26% higher among those participants with RHC use than those without RHC use.

TMLE

From the previous tutorial, we calculated the effective sample size for the outcome model as well as for the exposure model:

```
n <- nrow(dat)
pY <- nrow(dat[dat$Death == 1,])/n
n_eff <- min(n, 5*(n*min(pY, 1 - pY)))
n_eff
#> [1] 5735
```

```
n <- nrow(dat)
pA <- nrow(dat[dat$RHC.use == 1,])/n
n_eff <- min(n, 5*(n*min(pA, 1 - pA)))
n_eff
#> [1] 5735
```

Since the effective sample size is $\geq 5,000$ for both model, we can consider $5 \leq \mathrm{V} \leq 10$, where V is the number of folds. Let's work with 10-fold cross-validation for both the exposure and the outcome model, with the default super learner library.

```
fit.tmle <- tmle(Y = dat$Death,
                 A = dat$RHC.use,
                 W = dat[,baselinevars],
                 family = "binomial",
                 V.Q = 10,
                 V.g = 10)
```

```r
# OR
round(fit.tmle$estimates$OR$psi, 2)
#> [1] 1.26

# 95% CI
round(fit.tmle$estimates$OR$CI, 2)
#> [1] 1.12 1.42
```

As we can see that the odds of death was 26% higher among those participants with RHC use than those without RHC use.

Results comparison

Model	OR	95% CI
Logistic regression	1.42	1.32-1.65
Propensity score matching with logistic	1.29	1.12-1.48
Propensity score weighting with logistic	1.30	1.11-1.53
Propensity score matching with super learner (logistic, LASSO, and XGBoost)	1.26	1.09-1.45
Propensity score weighting with super learner (logistic, LASSO, and XGBoost)	1.26	1.04-1.52
TMLE (default SL library)	1.26	1.12-1.42

Figure 1: Forest Plot of Odds Ratios

Quiz (C)

> ⚠ Warning
>
> We are currently in the beta testing phase for our quiz page updates, so please be aware that some responses may not be accurate. Thank you for your patience.
> For those interested in downloading the quiz questions, you'll find a section below dedicated to downloading.

Live Quiz

You can click on the live quiz options to either see hints of why your response may be incorrect (response marked in red), or whether your choice of response is correct (response marked in green):

Download Quiz

1. **Downloading the File:**
   - Navigate to the link: See here.
   - Right-click on the link and select "Save link as…" from the dropdown menu.
   - Alternatively click here for downloading the quizzes.
   - Choose a destination folder on your computer where you'd like to save the file (e.g., Desktop). **Remember this location**, as you'll need to navigate to it later.
2. **Setting Up RStudio:**

- If you don't have RStudio installed, see the download link in here.
- Launch RStudio after installing.

3. **Installing Necessary R Packages:**

   - Before running the R Markdown file, ensure you have the required packages. In RStudio's console (located at the bottom left by default), enter the following commands to install them:

```r
install.packages("learnr")
install.packages("xfun")
```

4. **Opening and Running the File:**

   - In RStudio, go to `File` > `Open File...`.
   - Navigate to the folder where you saved the `Rmd` file and select it to open.
   - Once the file is open in RStudio, you'll see a "Run Document" button (green) at the top of the script editor. Click on it to run the R Markdown Quiz file.

R functions (C)

The list of new R functions introduced in this *Machine learning in causal inference* lab component are below:

Function.name	Package.name	Use
publish	Publish	To publish regression models
median	stats	To calculate the median of a numeric vector
SL.mean	SuperLearner	SuperLearner wrapper for the mean learner
SL.glm	SuperLearner	SuperLearner wrapper for the generalized linear model learner
SL.glmnet	SuperLearner	SuperLearner wrapper for the generalized linear model with elastic net penal
SL.xgboost	SuperLearner	SuperLearner wrapper for the extreme gradient boosting learner
SL.randomForest	SuperLearner	SuperLearner wrapper for the random forest learner
SL.svm	SuperLearner	SuperLearner wrapper for the support vector machine (SVM) learner
CreateTableOne	tableone	To create a summary table for a dataset
tmle	tmle	To run Targeted Maximum Likelihood Estimation (TMLE) for causal inferen
tmle.SL.dbarts2	tmle	SuperLearner wrapper for the Bayesian Additive Regression Trees (BART) le

Part XI

Complex outcomes

Background

This chapter offers tutorials on handling non-binary outcomes in data analysis. The first tutorial, on Polytomous and Ordinal outcomes, delves into multinomial and ordinal logistic regressions, teaching methods to analyze datasets with more than two categorical outcomes and how to assess the fit of ordinal logistic models. The second tutorial introduces Survival Analysis, guiding readers through the components of survival data, right censoring, the Kaplan-Meier method, the Cox regression model, and the complexities of analyzing time-dependent covariates and survey data. Lastly, the tutorial on Poisson and Negative Binomial focuses on regressions related to count data, showcasing both regular and survey-weighted methods, and highlights statistical techniques to understand factors influencing certain variables.

> **!** Important
>
> **Datasets**:
> All of the datasets used in this tutorial can be accessed from this GitHub repository folder

Overview of tutorials

Polytomous and ordinal

The tutorial covers methods for analyzing polytomous and ordinal outcomes in R. Initially, it introduces a function to exclude invalid responses from datasets. The data is loaded, and its structure is checked. The tutorial delves into the multinomial logistic regression, where tables are created to explore data characteristics. Two models are fitted: one for unweighted data and another for survey-weighted data. It then transitions into ordinal regression. Here, outcomes are ordered, and ordinal logistic models are fitted, again for both unweighted and survey-weighted data. The final part of the tutorial assesses the fit of the ordinal logistic model using various variables.

In the preceding chapters, we have primarily focused on binary or continuous outcomes. In this chapter, we will discuss about categorical [outcome variable is categorical with more than two categories (i.e., multicategory)], survival [time-to-event outcomes and aims to model the time until an event happens, accounting for censored data (observations where the event has not occurred by the end of the study)] and count data [non-negative integers (i.e., 0, 1, 2, 3, …)] outcomes.

Survival analysis

The tutorial provides an in-depth understanding of survival analysis. It starts by introducing the `lung` dataset which contains various attributes related to patients' health. The concept of censoring, specifically right censoring, is detailed, which happens when a subject leaves the study without experiencing the event of interest. The tutorial delves into the components of survival data, introducing the event indicator, survival function, and survival probability. It then transitions to practical applications, demonstrating how to create survival objects, estimate survival curves using the Kaplan-Meier method, and visualize these curves. There's also an emphasis on understanding and estimating median survival time and comparing survival times between different groups using the log-rank test. The Cox regression model, which is pivotal in survival analysis, is elaborated, along with the idea of hazard ratios and assessing proportional hazards. Time-dependent covariates, which account for variables that may change over time, are also touched upon. Towards the end, the tutorial addresses how to perform survival analysis on complex survey data, covering design creation, Kaplan-Meier plotting, and the Cox Proportional Hazards model in the context of these surveys.

Survival analysis in NHANES

The tutorial demonstrates an analysis of NHANES data linked with mortality data, and conducts a survival analysis.

Poisson and negative binomial

The tutorial provides a comprehensive guide on statistical analysis techniques related to Poisson and negative binomial regressions. Initially, it prepares the data by converting them into appropriate formats. A weighted summary of the dataset is then generated, emphasizing the distribution of a variable. The tutorial proceeds to demonstrate both regular and survey-weighted Poisson regressions, focusing on the relationship between fruit

consumption and various factors. Finally, it explores negative binomial regressions and their survey-weighted counterparts.

> Tip
>
> **Optional Content**:
> You'll find that some sections conclude with an optional video walkthrough that demonstrates the code. Keep in mind that the content might have been updated since these videos were recorded. Watching these videos is optional.

> Warning
>
> **Bug Report**:
> Fill out this form to report any issues with the tutorial.

Concepts (N)

Complex outcomes beyond binary

In this section, we explore regressions for outcomes beyond binary:

- Polytomous regression
- Survival analysis
- Poisson regression

Reading list

Key reference: (Hosmer, Lemeshow, and Sturdivant 2013; Kleinbaum and Klein 2011; Coxe, West, and Aiken 2009)

Video Lessons

Polytomous regression

Survival analysis

Poisson regression

Video Lesson Slides

Links

Polytomous regression

- Google Slides
- PDF Slides

Survival analysis

- Google Slides
- PDF Slides

Poisson regression

- Google Slides
- PDF Slides

References

Polytomous and ordinal

Let us load the packages:

```
# Load required packages
require(Publish)
require(survey)
require(svyVGAM)
require(car)
library(knitr)
```

In the code chunk below, we create a function called `invalid.exclude`. This function can be used to exclude invalid responses from the dataset, where don't know, refusal, and not stated are considered invalid responses.

```
# Function to exclude invalid responses
invalid.exclude <- function(Data, Var){
  subset.data <- subset(Data, eval(parse(text = Var)) != "DON'T KNOW" &
                          eval(parse(text = Var)) != "REFUSAL" &
                          eval(parse(text = Var)) != "NOT STATED")
  x1 <- dim(Data)[1]
  x2 <- dim(subset.data)[1]
  cat( format(x1-x2, big.mark = ","),
       "subjects deleted, and current N =" , format(x2, big.mark = ",") , "\n")
  return(subset.data)
}
```

Data

```
# Data
analytic <- readRDS("Data/nonbinary/cmh.Rds")
str(analytic)
```

```
#> 'data.frame':    2628 obs. of  9 variables:
#>  $ MHcondition       : Factor w/ 3 levels "Good","Poor or Fair",..: 2 2 3 3 3 2 3 1 3 3 ...
#>  $ CommunityBelonging: Factor w/ 4 levels "SOMEWHAT STRONG",..: 1 1 3 3 3 2 1 1 3 1 ...
#>  $ Age               : Factor w/ 6 levels "15 TO 24 YEARS",..: 4 1 1 4 3 5 2 5 3 1 ...
#>  $ Sex               : Factor w/ 2 levels "FEMALE","MALE": 1 1 2 2 2 2 2 1 1 2 ...
#>  $ RaceEthnicity     : Factor w/ 2 levels "NON-WHITE","WHITE": 2 1 2 2 1 2 2 2 2 1 ...
#>  $ MainIncome        : Factor w/ 5 levels "EI/WORKER'S COMP",..: 2 2 2 2 2 5 2 2 2 2 ...
#>  $ ReceivedHelp      : Factor w/ 4 levels "DON'T KNOW","NO",..: 4 4 4 2 2 4 2 2 4 2 ...
#>  $ Weight            : num  678 1298 196 917 2384 ...
#>  $ Disorder          : Factor w/ 1 level "YES": 1 1 1 1 1 1 1 1 1 1 ...
```

Let us drop invalid responses

```
# Drop invalid responses
analytic <- invalid.exclude(analytic, Var = c("CommunityBelonging", "Age", "Sex", "RaceEthnici
                                              "MainIncome", "MHcondition"))
#> 0 subjects deleted, and current N = 2,628
```

Multinomial logistic

Unweighted Tables

Let us see the summary statistic of the variables, stratified by mental health condition:

```
require("tableone")
tab1 <- CreateTableOne(vars = c("CommunityBelonging", "Age", "Sex", "RaceEthnicity", "MainIncom
                       strata = "MHcondition", data = analytic, test = F)
print(tab1, showAllLevels = TRUE, smd = T)
#>                          Stratified by MHcondition
#>                           level             Good        Poor or Fair
#>   n                                         885         1002
#>   CommunityBelonging (%) SOMEWHAT STRONG   362 (40.9)   288 (28.7)
#>                           SOMEWHAT WEAK     309 (34.9)   358 (35.7)
#>                           VERY STRONG        96 (10.8)    74 ( 7.4)
#>                           VERY WEAK         118 (13.3)   282 (28.1)
#>   Age (%)                 15 TO 24 YEARS    264 (29.8)   191 (19.1)
#>                           25 TO 34 YEARS    167 (18.9)   141 (14.1)
#>                           35 TO 44 YEARS    119 (13.4)   185 (18.5)
```

```
#>                          35 TO 54 YEARS    139 (15.7)   220 (22.0)
#>                          55 TO 64 YEARS    113 (12.8)   198 (19.8)
#>                          65 years or older  83 ( 9.4)    67 ( 6.7)
#>   Sex (%)                FEMALE            487 (55.0)   616 (61.5)
#>                          MALE              398 (45.0)   386 (38.5)
#>   RaceEthnicity (%)      NON-WHITE         140 (15.8)   184 (18.4)
#>                          WHITE             745 (84.2)   818 (81.6)
#>   MainIncome (%)         EI/WORKER'S COMP   78 ( 8.8)   195 (19.5)
#>                          EMPLOYMENT INC.   641 (72.4)   560 (55.9)
#>                          NOT STATED         23 ( 2.6)    25 ( 2.5)
#>                          OTHER              36 ( 4.1)    66 ( 6.6)
#>                          SENIOR BENEFITS   107 (12.1)   156 (15.6)
#>                         Stratified by MHcondition
#>                          Very good/excellent SMD
#>   n                      741
#>   CommunityBelonging (%) 355 (47.9)           0.425
#>                          190 (25.6)
#>                          116 (15.7)
#>                           80 (10.8)
#>   Age (%)                285 (38.5)           0.420
#>                          167 (22.5)
#>                           89 (12.0)
#>                           79 (10.7)
#>                           68 ( 9.2)
#>                           53 ( 7.2)
#>   Sex (%)                304 (41.0)           0.277
#>                          437 (59.0)
#>   RaceEthnicity (%)      134 (18.1)           0.045
#>                          607 (81.9)
#>   MainIncome (%)          36 ( 4.9)           0.400
#>                          598 (80.7)
#>                            9 ( 1.2)
#>                           22 ( 3.0)
#>                           76 (10.3)
```

Multinomial Model fitting

Before fitting the multinomial regression, let us redefine the reference categories of the variables.

```r
analytic$MHcondition2 <- relevel(analytic$MHcondition, ref = "Poor or Fair")
analytic$CommunityBelonging2 <- relevel(analytic$CommunityBelonging, ref = "VERY WEAK")
analytic$Age2 <- relevel(analytic$Age, ref = "65 years or older")
analytic$Sex2 <- relevel(analytic$Sex, ref = "FEMALE")
analytic$RaceEthnicity2 <- relevel(analytic$RaceEthnicity, ref = "NON-WHITE")
```

Now we will fit the multinomial logistic regression model using the `multinom` function from the `nnet` package:

```r
require(nnet)
fit4 <- multinom(MHcondition2 ~ CommunityBelonging2 + Sex2 + Age2 + RaceEthnicity2 + MainIncome
                    data = analytic)
#> # weights:  48 (30 variable)
#> initial  value 2887.153095
#> iter  10 value 2640.386436
#> iter  20 value 2625.127331
#> iter  30 value 2622.465409
#> final  value 2622.328038
#> converged
kable(round(exp(cbind(coef(fit4), confint(fit4))),2))
#> Warning in cbind(coef(fit4), confint(fit4)): number of rows of result is not a
#> multiple of vector length (arg 2)
```


```r
require(survey)
# Design
svy.analytic <- svydesign(ids = ~ 1, weights = ~ Weight, data = analytic)

# Table 1
tab1a <- svyCreateTableOne(vars = c("CommunityBelonging", "Age", "Sex", "RaceEthnicity", "Mainl
                  data = svy.analytic)
print(tab1a, showAllLevels = TRUE)
#>
#>                          level              Overall
#>   n                                         2734564.0
#>   CommunityBelonging (%) SOMEWHAT STRONG   1015381.0 (37.1)
#>                          SOMEWHAT WEAK      948838.7 (34.7)
#>                          VERY STRONG        297141.6 (10.9)
#>                          VERY WEAK          473202.7 (17.3)
#>   Age (%)                15 TO 24 YEARS     795568.6 (29.1)
#>                          25 TO 34 YEARS     564720.6 (20.7)
#>                          35 TO 44 YEARS     457821.1 (16.7)
#>                          35 TO 54 YEARS     447324.8 (16.4)
#>                          55 TO 64 YEARS     319374.3 (11.7)
#>                          65 years or older  149754.6 ( 5.5)
#>   Sex (%)                FEMALE            1312605.0 (48.0)
#>                          MALE              1421958.9 (52.0)
#>   RaceEthnicity (%)      NON-WHITE          565784.5 (20.7)
#>                          WHITE             2168779.4 (79.3)
#>   MainIncome (%)         EI/WORKER'S COMP   245758.5 ( 9.0)
#>                          EMPLOYMENT INC.   2059689.1 (75.3)
#>                          NOT STATED          60101.5 ( 2.2)
#>                          OTHER              116260.6 ( 4.3)
#>                          SENIOR BENEFITS    252754.4 ( 9.2)

# table 1 stratified by MHcondition
tab1b <- svyCreateTableOne(vars = c("CommunityBelonging", "Age", "Sex", "RaceEthnicity", "Mainl
                           strata = "MHcondition", data = svy.analytic)
print(tab1b, showAllLevels = TRUE)
#>                          Stratified by MHcondition
#>                           level              Good              Poor or Fair
#>   n                                         949208.1          973112.1
#>   CommunityBelonging (%) SOMEWHAT STRONG   346216.2 (36.5)   273163.9 (28.1)
#>                          SOMEWHAT WEAK     378431.4 (39.9)   365731.0 (37.6)
```

```
#>                          VERY STRONG        97528.8 (10.3)   69438.0 ( 7.1)
#>                          VERY WEAK         127031.7 (13.4)  264779.1 (27.2)
#>   Age (%)                15 TO 24 YEARS    286103.0 (30.1)  187610.9 (19.3)
#>                          25 TO 34 YEARS    189084.8 (19.9)  184528.6 (19.0)
#>                          35 TO 44 YEARS    140796.4 (14.8)  199172.7 (20.5)
#>                          35 TO 54 YEARS    171968.1 (18.1)  194689.1 (20.0)
#>                          55 TO 64 YEARS    109114.5 (11.5)  148432.2 (15.3)
#>                          65 years or older  52141.3 ( 5.5)   58678.6 ( 6.0)
#>   Sex (%)                FEMALE            436830.4 (46.0)  582311.1 (59.8)
#>                          MALE              512377.7 (54.0)  390801.0 (40.2)
#>   RaceEthnicity (%)      NON-WHITE         167887.9 (17.7)  220525.5 (22.7)
#>                          WHITE             781320.2 (82.3)  752586.6 (77.3)
#>   MainIncome (%)         EI/WORKER'S COMP   66003.9 ( 7.0)  151095.4 (15.5)
#>                          EMPLOYMENT INC.   752208.0 (79.2)  608703.4 (62.6)
#>                          NOT STATED         23589.0 ( 2.5)   28371.0 ( 2.9)
#>                          OTHER              32247.9 ( 3.4)   68453.1 ( 7.0)
#>                          SENIOR BENEFITS    75159.3 ( 7.9)  116489.2 (12.0)
#>                         Stratified by MHcondition
#>                          Very good/excellent p      test
#>   n                      812243.7
#>   CommunityBelonging (%) 396000.8 (48.8)     <0.001
#>                          204676.3 (25.2)
#>                          130174.7 (16.0)
#>                           81391.9 (10.0)
#>   Age (%)                321854.6 (39.6)     <0.001
#>                          191107.2 (23.5)
#>                          117852.1 (14.5)
#>                           80667.6 ( 9.9)
#>                           61827.6 ( 7.6)
#>                           38934.6 ( 4.8)
#>   Sex (%)                293463.5 (36.1)     <0.001
#>                          518780.2 (63.9)
#>   RaceEthnicity (%)      177371.1 (21.8)      0.219
#>                          634872.7 (78.2)
#>   MainIncome (%)          28659.2 ( 3.5)     <0.001
#>                          698777.7 (86.0)
#>                            8141.5 ( 1.0)
#>                           15559.6 ( 1.9)
#>                           61105.8 ( 7.5)
```

Setting up the design

Let us redefine the reference categories within the survey design and convert the design to use replicate weights.

```
w.design <- svydesign(id=~1, weights=~Weight, data=analytic)
w.design <- update(w.design , MHcondition2 = relevel(MHcondition, ref = "Poor or Fair"),
                   CommunityBelonging2 = relevel(CommunityBelonging, ref = "VERY WEAK"),
                   Age2 = relevel(Age, ref = "65 years or older"),
                   Sex2 = relevel(Sex, ref = "FEMALE"),
                   RaceEthnicity2 = relevel(RaceEthnicity, ref = "NON-WHITE"))

# Convert a survey design to use replicate weights
w.design2 <- as.svrepdesign(w.design, type = "bootstrap" , replicates = 50)
```

Multinomial Model fitting

Now, we will fit the design-adjusted multinomial logistic regression:

```
require(svyVGAM)
fit5 <- svy_vglm(MHcondition2 ~ CommunityBelonging2 + Sex2 + Age2 + RaceEthnicity2 + MainIncome
                     design = w.design2, family = multinomial)
```

```
kable(round(exp(cbind(coef(fit5), confint(fit5))),2))
```

		2.5 %	97.5 %
(Intercept):1	16.57	5.21	52.70
(Intercept):2	3.24	1.20	8.81
CommunityBelonging2SOMEWHAT STRONG:1	0.22	0.13	0.37
CommunityBelonging2SOMEWHAT STRONG:2	0.56	0.37	0.85
CommunityBelonging2SOMEWHAT WEAK:1	0.58	0.36	0.92
CommunityBelonging2SOMEWHAT WEAK:2	1.18	0.73	1.89
CommunityBelonging2VERY STRONG:1	0.15	0.08	0.28
CommunityBelonging2VERY STRONG:2	0.46	0.25	0.84
Sex2MALE:1	0.39	0.27	0.54
Sex2MALE:2	0.68	0.48	0.96
Age215 TO 24 YEARS:1	0.56	0.25	1.21

		2.5 %	97.5 %
Age215 TO 24 YEARS:2	0.65	0.33	1.27
Age225 TO 34 YEARS:1	0.83	0.37	1.89
Age225 TO 34 YEARS:2	0.68	0.31	1.49
Age235 TO 44 YEARS:1	1.74	0.77	3.91
Age235 TO 44 YEARS:2	0.91	0.43	1.91
Age235 TO 54 YEARS:1	2.02	0.84	4.84
Age235 TO 54 YEARS:2	1.48	0.65	3.35
Age255 TO 64 YEARS:1	1.91	0.83	4.37
Age255 TO 64 YEARS:2	1.33	0.60	2.90
RaceEthnicity2WHITE:1	0.91	0.63	1.30
RaceEthnicity2WHITE:2	1.22	0.88	1.70
MainIncomeEMPLOYMENT INC.:1	0.25	0.13	0.46
MainIncomeEMPLOYMENT INC.:2	0.58	0.31	1.11
MainIncomeNOT STATED:1	0.91	0.21	4.01
MainIncomeNOT STATED:2	1.37	0.36	5.22
MainIncomeOTHER:1	1.43	0.58	3.57
MainIncomeOTHER:2	1.30	0.50	3.40
MainIncomeSENIOR BENEFITS:1	0.39	0.17	0.86
MainIncomeSENIOR BENEFITS:2	0.49	0.24	0.99

Ordinal Regression

Ordering outcome

Let's define `MHcondition` as an ordinal outcome. We can do it by using `ordered = TRUE` in the `factor` function.

```
# Ordinal outcome
analytic$MHcondition3 <- factor(analytic$MHcondition,
                                levels = c("Poor or Fair", "Good", "Very good or excellent"),
                        ordered = TRUE)
```

Ordinal logistic

Let's fit the ordinal logistic using the `polr` function:

```r
require(MASS)
fit5o1 <- polr(MHcondition3 ~CommunityBelonging2 +
                  Sex2 + Age2 + RaceEthnicity2 + MainIncome,
                data=analytic)
kable(round(exp(cbind(coef(fit5o1), confint(fit5o1))),2))
#> Waiting for profiling to be done...
#>
#> Re-fitting to get Hessian
```

		2.5 %	97.5 %
CommunityBelonging2SOMEWHAT STRONG	2.69	2.05	3.55
CommunityBelonging2SOMEWHAT WEAK	1.83	1.40	2.41
CommunityBelonging2VERY STRONG	3.15	2.15	4.66
Sex2MALE	1.30	1.07	1.58
Age215 TO 24 YEARS	0.69	0.42	1.11
Age225 TO 34 YEARS	0.60	0.36	0.98
Age235 TO 44 YEARS	0.32	0.19	0.53
Age235 TO 54 YEARS	0.37	0.23	0.59
Age255 TO 64 YEARS	0.36	0.22	0.56
RaceEthnicity2WHITE	1.26	0.97	1.63
MainIncomeEMPLOYMENT INC.	2.26	1.68	3.06
MainIncomeNOT STATED	1.53	0.79	2.94
MainIncomeOTHER	1.15	0.70	1.90
MainIncomeSENIOR BENEFITS	1.10	0.70	1.70

Ordinal logistic for complex survey

The same as before, we can set up the design, relevel variables within the design, define MHcondition as an ordinal variable, and then fit the design-adjusted ordinal logistic regression.

```r
w.design <- svydesign(id=~1, weights=~Weight, data=analytic)
w.design<-update(w.design ,
                  CommunityBelonging2=relevel(CommunityBelonging, ref="VERY WEAK"),
                  Age2=relevel(Age, ref="65 years or older"),
                  Sex2=relevel(Sex, ref="FEMALE"),
                  RaceEthnicity2=relevel(RaceEthnicity, ref="NON-WHITE"),
                  MHcondition3 = factor(MHcondition, levels=c("Poor or Fair", "Good",
```

```r
                                                                    "Very good or excellent"),
                                                ordered=TRUE))

# Design-adjusted Ordinal logistic
fit5o <- svyolr(MHcondition3 ~CommunityBelonging2 + Sex2 + Age2 + RaceEthnicity2 + MainIncome,
                design=w.design)
kable(round(exp(cbind(coef(fit5o), confint(fit5o))),2))
```

		2.5 %	97.5 %
CommunityBelonging2SOMEWHAT STRONG	2.49	1.65	3.75
CommunityBelonging2SOMEWHAT WEAK	2.04	1.37	3.05
CommunityBelonging2VERY STRONG	3.02	1.74	5.26
Sex2MALE	1.76	1.30	2.38
Age215 TO 24 YEARS	1.13	0.52	2.44
Age225 TO 34 YEARS	0.77	0.34	1.74
Age235 TO 44 YEARS	0.52	0.24	1.13
Age235 TO 54 YEARS	0.70	0.32	1.52
Age255 TO 64 YEARS	0.69	0.32	1.46
RaceEthnicity2WHITE	1.31	0.90	1.91
MainIncomeEMPLOYMENT INC.	2.37	1.53	3.67
MainIncomeNOT STATED	1.42	0.52	3.86
MainIncomeOTHER	0.88	0.39	2.00
MainIncomeSENIOR BENEFITS	1.26	0.66	2.41
Poor or Fair|Good	4.79	1.92	11.91
Good|Very good or excellent	2636603.90	996905.04	6973262.13

Assessing model fit

We can use the `regTermTest` function from the `survey` package to do the Wald test of a regression coefficient.

```r
regTermTest(fit5o, ~CommunityBelonging2 , df = Inf)
#> Wald test for CommunityBelonging2
#>  in svyolr(MHcondition3 ~ CommunityBelonging2 + Sex2 + Age2 + RaceEthnicity2 +
#>     MainIncome, design = w.design)
#> Chisq =  24.53941  on  3  df: p= 1.9272e-05
regTermTest(fit5o, ~Age2 , df = Inf)
#> Wald test for Age2
```

```
#>  in svyolr(MHcondition3 ~ CommunityBelonging2 + Sex2 + Age2 + RaceEthnicity2 +
#>     MainIncome, design = w.design)
#> Chisq =  14.00491  on  5  df: p= 0.015578
regTermTest(fit5o, ~Sex2 , df = Inf)
#> Wald test for Sex2
#>  in svyolr(MHcondition3 ~ CommunityBelonging2 + Sex2 + Age2 + RaceEthnicity2 +
#>     MainIncome, design = w.design)
#> Chisq =  13.46032  on  1  df: p= 0.00024366
regTermTest(fit5o, ~RaceEthnicity2 , df = Inf)
#> Wald test for RaceEthnicity2
#>  in svyolr(MHcondition3 ~ CommunityBelonging2 + Sex2 + Age2 + RaceEthnicity2 +
#>     MainIncome, design = w.design)
#> Chisq =  2.002794  on  1  df: p= 0.15701
regTermTest(fit5o, ~MainIncome , df = Inf)
#> Wald test for MainIncome
#>  in svyolr(MHcondition3 ~ CommunityBelonging2 + Sex2 + Age2 + RaceEthnicity2 +
#>     MainIncome, design = w.design)
#> Chisq =  22.13656  on  4  df: p= 0.00018826
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Survival analysis

```r
# Load required packages
require(knitr)
require(Publish)
require(survey)
library(broom)
library(tidyverse)
require(survminer)
```

The lung dataset

- inst: Institution code
- time: Survival time in days
- status: censoring status
  - 1=censored,
  - 2=dead
- age: Age in years
- sex:
  - Male=1
  - Female=2
- ph.ecog: ECOG performance score (0=good 5=dead)
- ph.karno: Karnofsky performance score (bad=0-good=100) rated by physician
- pat.karno: Karnofsky performance score as rated by patient
- meal.cal: Calories consumed at meals
- wt.loss: Weight loss in last six months

```r
library(survival)
head(lung)
#>   inst time status age sex ph.ecog ph.karno pat.karno meal.cal wt.loss
#> 1    3  306      2  74   1       1       90       100     1175      NA
#> 2    3  455      2  68   1       0       90        90     1225      15
#> 3    3 1010      1  56   1       0       90        90       NA      15
#> 4    5  210      2  57   1       1       90        60     1150      11
#> 5    1  883      2  60   1       0      100        90       NA       0
#> 6   12 1022      1  74   1       1       50        80      513       0
kable(head(lung))
```

inst	time	status	age	sex	ph.ecog	ph.karno	pat.karno	meal.cal	wt.loss
3	306	2	74	1	1	90	100	1175	NA
3	455	2	68	1	0	90	90	1225	15
3	1010	1	56	1	0	90	90	NA	15
5	210	2	57	1	1	90	60	1150	11
1	883	2	60	1	0	100	90	NA	0
12	1022	1	74	1	1	50	80	513	0

```r
# kable(lung)
```

What is censoring?

Censoring occurs if a subject leave the study without experiencing the event

Types of censoring

Left censoring:

The subject has experienced the event before enrollment

Right censoring:

- Loss to follow-up
- Withdraw from the study

- Survive to the end of the study

In this lab, we focus on right censoring

Components of survival data

There are four components in survival data:

For any individual i:

- Event time T_i
- Censoring time C_i
- Event indicator δ_i
- Observed time $Y_i = min(T_i, C_i)$

Event indicator

The event indicator δ_i was defined as a binary data

- $\delta_i = 1$ if $T_i \leq C_i$
- $\delta_i = 0$ if $T_i > C_i$

Survival function

Survival function represents the probability that an individual will survival past time t

$S(t) = Pr(T > t) = 1 - F(t)$, where $F(t)$ is the cumulative distribution function.

Survival probability

Given that a subject is still alive just before time t, the survival probability $S(t)$ represents the probability of surviving beyond time t. We could plot survival probability in lung dataset

Creating survival objects

A survival object is the survival time and can be used as the "response" in survival analysis. To estimate $S(t)$, we usually use the non-parametric method called Kaplan-Meier method. In R, we can use "Surv" in survival package to create survival objects. Let us take a look at the first 10 subjects in lung dataset. The numbers are the survival time for each individual, + indicates that the observation has been censored

```r
Surv(lung$time, lung$status)[1:10]
#>  [1]  306   455  1010+  210   883  1022+  310   361   218   166
```

Estimating survival curves with the Kaplan-Meier method

"survfit" in survival package can help us to estimate survival curves using KM method

```r
f1 <- survfit(Surv(time, status) ~ 1, data = lung)
# Surv(time, status) is the survival object we just introduced
# show the objects in the f1
names(f1)
#>  [1] "n"         "time"      "n.risk"    "n.event"   "n.censor"  "surv"
#>  [7] "std.err"   "cumhaz"    "std.chaz"  "type"      "logse"     "conf.int"
#> [13] "conf.type" "lower"     "upper"     "call"
```

Kaplan-Meier plot

We could plot the above fitting to have a KM plot in R

```r
plot(survfit(Surv(time, status) ~ 1, data = lung),
     xlab = "Days",
     ylab = "Overall survival probability")
```

```
## We could also do KM by groups
plot(survfit(Surv(time, status) ~ sex, data = lung),
     xlab = "Days",
     ylab = "Overall survival probability")
```

```
## Adding legend
fit0 <- survfit(Surv(time, status) ~ sex, data = lung)
plot(fit0,
     xlab = "Days",
```

```r
     ylab = "Overall survival probability",
     lty = 1:2,col=1:2)
Lab.x <- names(fit0$strata)
legend("topright", legend=Lab.x,
  col=1:2, lty=1:2,horiz=FALSE,bty='n')
```

```r
#library(survminer)
# ggsurvplot(survfit(Surv(time, status) ~ sex, data = lung), data = lung,
#            xlab="Days",
#            ylab = "Survival probability")
```

Estimating median survival time

Usually, the survival time will not be normally (or symmetrically) distributed. Therefore, mean is not a good summary for survival time. Instead, we use median to estimate. "survfit" in survival package present the summary of median

```r
survfit(Surv(time, status) ~ 1, data = lung)
#> Call: survfit(formula = Surv(time, status) ~ 1, data = lung)
#>
#>        n events median 0.95LCL 0.95UCL
#> [1,] 228    165    310     285     363
```

Comparing survival times between groups

We could use log-rank test to test whether there exists significant difference in survival time between groups. The log-rank test put the same weights on every observation. It could be done in R use "survdiff" from survival package

```
survdiff(Surv(time, status) ~ sex, data = lung)
#> Call:
#> survdiff(formula = Surv(time, status) ~ sex, data = lung)
#>
#>         N Observed Expected (O-E)^2/E (O-E)^2/V
#> sex=1 138      112     91.6      4.55      10.3
#> sex=2  90       53     73.4      5.68      10.3
#>
#>  Chisq= 10.3  on 1 degrees of freedom, p= 0.001
```

The results indicated that the survival time depends on the sex (p-value = 0.001); e.g., associated with sex variable.

The Cox regression model

The Cox regression model (aka Cox proportional-hazard model) is semi-parametric model to quantify the effect size in survival analysis.

- $h(t|X_i) = h_0(t)exp(\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \beta_3 X_{i3} + ... + \beta_p X_{ip})$, where
  - $h_0(t)$ is the baseline hazard and
  - $h(t)$ is the hazard at time t

In R, we have "coxph" from survival package to fit a Cox PH model

```
fit <- coxph(Surv(time, status) ~ sex, data = lung) # add sex as covariate
summary(fit)
#> Call:
#> coxph(formula = Surv(time, status) ~ sex, data = lung)
#>
```

```
#>   n= 228, number of events= 165
#>
#>        coef exp(coef) se(coef)      z Pr(>|z|)
#> sex -0.5310    0.5880   0.1672 -3.176  0.00149 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>     exp(coef) exp(-coef) lower .95 upper .95
#> sex     0.588      1.701    0.4237     0.816
#>
#> Concordance= 0.579  (se = 0.021 )
#> Likelihood ratio test= 10.63  on 1 df,   p=0.001
#> Wald test            = 10.09  on 1 df,   p=0.001
#> Score (logrank) test = 10.33  on 1 df,   p=0.001
require(Publish)
publish(fit)
#>  Variable Units HazardRatio       CI.95   p-value
#>       sex              0.59 [0.42;0.82]   0.00149
```

More variables

```
fit <- coxph(Surv(time, status) ~ sex + ph.ecog +
               ph.karno + pat.karno + meal.cal + wt.loss,
             data = lung) # add more covariates
summary(fit)
#> Call:
#> coxph(formula = Surv(time, status) ~ sex + ph.ecog + ph.karno +
#>     pat.karno + meal.cal + wt.loss, data = lung)
#>
#>   n= 168, number of events= 121
#>    (60 observations deleted due to missingness)
#>
#>                 coef  exp(coef)   se(coef)      z Pr(>|z|)
#> sex       -5.571e-01  5.729e-01  2.003e-01 -2.782  0.00541 **
#> ph.ecog    7.377e-01  2.091e+00  2.260e-01  3.264  0.00110 **
#> ph.karno   2.041e-02  1.021e+00  1.108e-02  1.842  0.06549 .
#> pat.karno -1.263e-02  9.874e-01  8.062e-03 -1.567  0.11707
#> meal.cal  -8.303e-06  1.000e+00  2.567e-04 -0.032  0.97420
#> wt.loss   -1.460e-02  9.855e-01  7.708e-03 -1.894  0.05820 .
#> ---
```

```
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>           exp(coef) exp(-coef) lower .95 upper .95
#> sex          0.5729     1.7456    0.3869    0.8483
#> ph.ecog      2.0910     0.4782    1.3427    3.2565
#> ph.karno     1.0206     0.9798    0.9987    1.0430
#> pat.karno    0.9874     1.0127    0.9720    1.0032
#> meal.cal     1.0000     1.0000    0.9995    1.0005
#> wt.loss      0.9855     1.0147    0.9707    1.0005
#>
#> Concordance= 0.656  (se = 0.029 )
#> Likelihood ratio test= 27.47  on 6 df,   p=1e-04
#> Wald test            = 27.02  on 6 df,   p=1e-04
#> Score (logrank) test = 27.82  on 6 df,   p=1e-04
```

Formatting Cox regression results

To format the Cox regression results into a nice table format, we could use "tidy" (broom package).

```r
library(broom)
library(tidyverse)
summary(fit)
#> Call:
#> coxph(formula = Surv(time, status) ~ sex + ph.ecog + ph.karno +
#>     pat.karno + meal.cal + wt.loss, data = lung)
#>
#>   n= 168, number of events= 121
#>    (60 observations deleted due to missingness)
#>
#>                 coef  exp(coef)   se(coef)      z Pr(>|z|)
#> sex       -5.571e-01  5.729e-01  2.003e-01 -2.782  0.00541 **
#> ph.ecog    7.377e-01  2.091e+00  2.260e-01  3.264  0.00110 **
#> ph.karno   2.041e-02  1.021e+00  1.108e-02  1.842  0.06549 .
#> pat.karno -1.263e-02  9.874e-01  8.062e-03 -1.567  0.11707
#> meal.cal  -8.303e-06  1.000e+00  2.567e-04 -0.032  0.97420
#> wt.loss   -1.460e-02  9.855e-01  7.708e-03 -1.894  0.05820 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#>
#>            exp(coef) exp(-coef) lower .95 upper .95
#> sex           0.5729     1.7456    0.3869    0.8483
#> ph.ecog       2.0910     0.4782    1.3427    3.2565
#> ph.karno      1.0206     0.9798    0.9987    1.0430
#> pat.karno     0.9874     1.0127    0.9720    1.0032
#> meal.cal      1.0000     1.0000    0.9995    1.0005
#> wt.loss       0.9855     1.0147    0.9707    1.0005
#>
#> Concordance= 0.656  (se = 0.029 )
#> Likelihood ratio test= 27.47  on 6 df,   p=1e-04
#> Wald test            = 27.02  on 6 df,   p=1e-04
#> Score (logrank) test = 27.82  on 6 df,   p=1e-04
require(Publish)
publish(fit)
#>   Variable Units Missing HazardRatio       CI.95   p-value
#>        sex             0        0.57 [0.39;0.85]   0.00541
#>    ph.ecog             1        2.09 [1.34;3.26]   0.00110
#>   ph.karno             1        1.02 [1.00;1.04]   0.06549
#>  pat.karno             3        0.99 [0.97;1.00]   0.11707
#>   meal.cal            47        1.00 [1.00;1.00]   0.97420
#>    wt.loss            14        0.99 [0.97;1.00]   0.05820
kable(broom::tidy(fit, exp = TRUE))
```

term	estimate	std.error	statistic	p.value
sex	0.5728725	0.2002798	-2.7815687	0.0054097
ph.ecog	2.0910352	0.2260265	3.2635959	0.0011001
ph.karno	1.0206185	0.0110803	1.8418903	0.0654912
pat.karno	0.9874450	0.0080618	-1.5672047	0.1170668
meal.cal	0.9999917	0.0002567	-0.0323431	0.9741984
wt.loss	0.9855065	0.0077076	-1.8941781	0.0582014

```r
kable(broom::tidy(fit, exp = TRUE), digits = 2)
```

term	estimate	std.error	statistic	p.value
sex	0.57	0.20	-2.78	0.01
ph.ecog	2.09	0.23	3.26	0.00

term	estimate	std.error	statistic	p.value
ph.karno	1.02	0.01	1.84	0.07
pat.karno	0.99	0.01	-1.57	0.12
meal.cal	1.00	0.00	-0.03	0.97
wt.loss	0.99	0.01	-1.89	0.06

```r
require(survminer)
ggforest(fit, data = lung)
```

Hazard ratios

- In Cox regression model, the quantity we interested is the **hazard ratio**, which is the ratio of hazards in two different groups (i.e., exposed vs unexposed).
- Hazard ratio at time t is denoted by $HR = \frac{h_1(t)}{h_0(t)}$ where $h(t)$ is the hazard function representing the instantaneous rate that the first event may occur.
- Hazard ratio is **not** a risk, and it can estimated by exponatial the estimated coefficients (i.e., $exp(\beta)$).
- In the above Cox model, the HR is 0.59 which indicated that the hazard in Female is smaller than Male.
- In generally, a $HR > 1$ indicates a higher hazard, and $HR < 1$ indicated a reduced hazard.

Assessing proportional hazards

- As mentioned before, the Cox model is also called as Cox proportional hazard model.
- The assumption made here is the hazards in both group are proportional.
- The easiest way of assessing proportional hazard assumption is to use "cox.zph"
- This function uses the Schoenfeld residuals against the transformed time.
- Ideally, this checks if the hazard rate of an individual is relatively constant in time.

```
f2 <- coxph(Surv(time, status) ~ sex, data = lung)
test <- cox.zph(f2)
test
#>        chisq df     p
#> sex     2.86  1 0.091
#> GLOBAL  2.86  1 0.091
plot(test)
```

- Here, p-value is greater than 0.05, there is no evidence against PH assumption.
- Having very small p values (say, 0.001) indicates that there may be time dependent coefficients which the modelling needs to take care of.

Time-dependent covariate

Time-dependent covariate occurs when individual's covariate values are may be different at different time (measured repeatedly over time).

Data setup

```r
# Create a simple data set for a time-dependent model
# See ?coxph
test.data <- list(start=c(1, 2, 5, 2, 1, 7, 3, 4, 8, 8),
                  stop =c(2, 3, 6, 7, 8, 9, 9, 9,14,17),
                  event=c(1, 1, 1, 1, 1, 1, 1, 0, 0, 0),
                  tx  =c(1, 0, 0, 1, 0, 1, 1, 1, 0, 0) )
test.data
#> $start
#>  [1] 1 2 5 2 1 7 3 4 8 8
#>
#> $stop
#>  [1]  2  3  6  7  8  9  9  9 14 17
#>
#> $event
#>  [1] 1 1 1 1 1 1 1 0 0 0
#>
#> $tx
#>  [1] 1 0 0 1 0 1 1 1 0 0
as.data.frame(test.data)
#>    start stop event tx
#> 1      1    2     1  1
#> 2      2    3     1  0
#> 3      5    6     1  0
#> 4      2    7     1  1
#> 5      1    8     1  0
#> 6      7    9     1  1
#> 7      3    9     1  1
#> 8      4    9     0  1
#> 9      8   14     0  0
#> 10     8   17     0  0
```

Time-dependent covariate - Cox regression

```
fit.td <- coxph( Surv(start, stop, event) ~ tx, test.data)
summary(fit.td)
#> Call:
#> coxph(formula = Surv(start, stop, event) ~ tx, data = test.data)
#>
#>   n= 10, number of events= 7
#>
#>        coef exp(coef) se(coef)      z Pr(>|z|)
#> tx -0.02111   0.97912  0.79518 -0.027    0.979
#>
#>    exp(coef) exp(-coef) lower .95 upper .95
#> tx    0.9791      1.021    0.2061     4.653
#>
#> Concordance= 0.526  (se = 0.129 )
#> Likelihood ratio test= 0  on 1 df,   p=1
#> Wald test            = 0  on 1 df,   p=1
#> Score (logrank) test = 0  on 1 df,   p=1
publish(fit.td)
#>  Variable Units HazardRatio       CI.95 p-value
#>        tx              0.98 [0.21;4.65]   0.979
```

Survival analysis in Complex Survey data

- Example data from GitHub.
- Below we created the design

Data and Variables

```
load("Data/nonbinary/nh_99-06.Rdata") #
analytic.miss <- as.data.frame(MainTable[c("SDMVPSU", "SDMVSTRA", "WTMEC2YR", "PERMTH_INT", "PI
```

- MORTSTAT: Final Mortality Status
  - 0 Assumed alive
  - 1 Assumed deceased

  - Blank Ineligible for mortality follow-up or under age 17

- PERMTH_EXM: Person Months of Follow-up from MEC/Home Exam Date
  - 0 - 217
  - Blank Ineligible

- PERMTH_INT: Person Months of Follow-up from Interview Date

Data issues

```
with(analytic.miss[1:30,],
     Surv(PERMTH_EXM, MORTSTAT))
#>  [1] NA? 90+ NA? NA? 74+ 86+ 76+ NA? NA? 79+ NA? 82+ 16  85+ 92+ 62  NA? NA? NA?
#> [20] 86+ 87+ NA? NA? 72+ 84+ NA? 85+ 91+ 26  NA?
summary(analytic.miss$WTMEC2YR)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>       0    6365   15572   27245   38896  261361
# avoiding 0 weight issues
analytic.miss$WTMEC2YR[analytic.miss$WTMEC2YR == 0] <- 0.001
require(DataExplorer)
plot_missing(analytic.miss)
```

Design creation

```
analytic.miss$ID <- 1:nrow(analytic.miss)
analytic.miss$miss <- 0
analytic.cc <- as.data.frame(na.omit(analytic.miss))
dim(analytic.cc)
#> [1] 10557    11
analytic.miss$miss[analytic.miss$ID %in%
                     analytic.cc$ID] <- 0
w.design0 <- svydesign(id=~SDMVPSU,
                       strata=~SDMVSTRA,
                       weights=~WTMEC2YR,
                       nest=TRUE,
                       data=analytic.miss)
w.design <- subset(w.design0,
                   miss == 0)
summary(weights(w.design))
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>       0    6365   15572   27245   38896  261361
```

Survival Analysis within Complex Survey

KM plot

```
fit0 <- svykm(Surv(PERMTH_EXM, MORTSTAT) ~ 1,
              design = w.design)
plot(fit0)
```

Cox PH

```
fit <- svycoxph(Surv(PERMTH_EXM, MORTSTAT) ~
                  white + female + RIDAGEYR,
                design = w.design)
publish(fit)
#> Stratified 1 - level Cluster Sampling design (with replacement)
#> With (117) clusters.
#> subset(w.design0, miss == 0)
#>  Variable Units HazardRatio       CI.95   p-value
#>     white              0.78 [0.66;0.93]   0.00441
#>    female              0.61 [0.50;0.75]   < 0.001
#>  RIDAGEYR              1.09 [1.08;1.10]   < 0.001
```

PH assumption

```
testPh <- cox.zph(fit)
print(testPh)
#>              chisq df    p
#> white    7.39e-05  1 0.99
#> female   2.84e-07  1 1.00
#> RIDAGEYR 9.16e-05  1 0.99
```

```r
#> GLOBAL   1.45e-04  3 1.00
plot(testPh)
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Survival analysis: NHANES

The tutorial demonstrates analyzing complex survey data (NHANES) with mortality as a survival outcome. See the tutorial in the previous chapter on linking public-use US mortality data with the NHANES.

We will explore the relationship between caffeine consumption and mortality in adults with diabetes using NHANES 1999-2010 datasets. We will follow the following article by Neves et al. (2018).

Load required packages

```
# Load required packages
library(tableone)
library(survival)
library(Publish)
library(survey)
library(DataExplorer)
```

Load datasets

Let us load NHANES 1999-2000, 2001-2002, 2003-2004, 2005-2006, 2007-2008, and 2009-2010 datasets and merged all cycles.

```
# Load
load("Data/nonbinary/coffee.RData")
ls()
#> [1] "dat.analytic1999" "dat.analytic2001" "dat.analytic2003" "dat.analytic2005"
#> [5] "dat.analytic2007" "dat.analytic2009" "has_annotations"

```

```r
# Merge all cycles
dat.full <- rbind(dat.analytic1999, dat.analytic2001, dat.analytic2003,
                  dat.analytic2005, dat.analytic2007, dat.analytic2009)
head(dat.full)
#>   id survey.weight psu strata           caff stime status    sex age
#> 1  1      9727.079   1      5 No consumption    NA     NA Female   2
#> 2  2     26678.636   3      1    200+ mg/day   177      1   Male  77
#> 3  3     43621.681   2      7    <100 mg/day    NA     NA Female  10
#> 4  4     10346.119   1      2 No consumption    NA     NA   Male   1
#> 5  5     91050.847   2      8 No consumption   244      0   Male  49
#> 6  6     36508.250   2      2    200+ mg/day   246      0 Female  19
#>                 race             income       smoking diabetic.nephropathy
#> 1 Non-Hispanic Black           <$25,000          <NA>                   NA
#> 2 Non-Hispanic White $25,000 to $75,000  Never smoker                   NA
#> 3 Non-Hispanic White $25,000 to $75,000          <NA>                   NA
#> 4 Non-Hispanic Black           <$25,000          <NA>                   NA
#> 5 Non-Hispanic White           >$75,000 Former smoker                   NA
#> 6         Other race           <$25,000          <NA>                   NA
#>         bmi.cat                 education carbohyd        alcohol  diab.years
#> 1         <20.0                      <NA>   250.36 No consumption Undiagnosed
#> 2 20.0 to <25.0 College graduate or above   350.37 No consumption Undiagnosed
#> 3         <20.0                      <NA>   233.63 No consumption Undiagnosed
#> 4          <NA>                      <NA>   191.03 No consumption Undiagnosed
#> 5 25.0 to <30.0 College graduate or above   253.98  20+ grams/day Undiagnosed
#> 6 20.0 to <25.0                      <NA>   202.11 No consumption Undiagnosed
#>    htn daib.retino macrovascular insulin survey.cycle physical.activity
#> 1 <NA>        <NA>          <NA>    <NA>      1999-00              <NA>
#> 2   No        <NA>            No    <NA>      1999-00              High
#> 3 <NA>        <NA>          <NA>      No      1999-00              <NA>
#> 4 <NA>        <NA>          <NA>    <NA>      1999-00              <NA>
#> 5  Yes        <NA>            No      No      1999-00              High
#> 6   No        <NA>          <NA>    <NA>      1999-00      Intermediate
#>   diabetes cal.total
#> 1       No   1358.88
#> 2       No   2463.00
#> 3       No   1517.69
#> 4       No   1474.93
#> 5       No   2658.14
#> 6       No   1113.66
dim(dat.full)
```

```
#> [1] 62160    26
```

The merged dataset contains 62,160 subjects with 26 relevant variables:

- id: Respondent sequence/ID number
- survey.weight: Full sample 2 year weights
- psu: Masked pseudo-PSU
- strata: Masked pseudo-stratum
- caff: Caffeine (exposure variable)
- stime: Follow-up time (time from interview date to death or censoring)
- status: Mortality status
- sex: Sex
- age: Age in years
- race: Race/ethnicity
- income: Annual household income
- smoking: Smoking status
- diabetic.nephropathy: Diabetic nephropathy (no data)
- bmi.cat: BMI - categorical
- education: Education level
- carbohyd: Carbohydrate in gm
- alcohol: Alcohol consumption
- diab.years: Years since diabetes
- htn: Hypertension
- daib.retino: Diabetes retinopathy
- macrovascular: Macrovascular complications
- insulin: Insulin
- survey.cycle: Survey cycle
- physical.activity: Physical activity
- diabetes: Diabetes status
- cal.total: Total calories in kcal

Data pre-processing

Eligibility criteria

The authors considered adults aged 18 years or more, only diabetic, and total calories between 500 and 3500.

```r
# Total samples in the merged dataset
nrow(dat.full) # N = 62,160
#> [1] 62160

# Age >= 18 years
dat2 <- subset(dat.full, age >= 18)
nrow(dat2) # N = 35,379
#> [1] 35379

# With diabetes
dat3 <- subset(dat2, diabetes == "Yes")
nrow(dat3) # N = 4,687 - numbers don't match with the paper (N = 4,544)
#> [1] 4687

# Implausible alimentary reports
dat4 <- subset(dat3, cal.total >= 500 & cal.total <= 3500)
nrow(dat4) # N = 4,083 - numbers don't match with the paper (N = 3,948)
#> [1] 4083
```

Complete case data

Let us drop missing values in the exposure and outcome:

```r
# Drop missing exposure and outcome
dat <- dat4[complete.cases(dat4$id),]
dat <- dat[complete.cases(dat$caff),]
dat <- dat[complete.cases(dat$status),]
dat <- dat[complete.cases(dat$stime),] # N = 4,080
dim(dat)
#> [1] 4080   26

# Missing plot
plot_missing(dat)
```

Now, let us drop variables with high missingness for this exercise. As explained in the Missing data analysis chapter, a better approach could be imputing missing values under the missing at random assumption.

```
# Drop variables with high missingness
dat$diabetic.nephropathy <- dat$diab.years <- dat$daib.retino <- dat$income <- NULL

# Complete case data
dat <- na.omit(dat)
dim(dat) # N = 3,780
#> [1] 3780   22
```

Table 1

Now, let us create Table 1 stratified by coffee consumption (exposure), separately for males and females, as done in the article.

```
vars <- c("age", "race", "education", "smoking", "alcohol", "carbohyd", "physical.activity",
          "bmi.cat", "htn", "macrovascular", "insulin", "survey.cycle")

tab1a <- CreateTableOne(vars = vars, strata = "caff", data = dat[dat$sex=="Female",],
                        test = F)
```

```r
tab1b <- CreateTableOne(vars = vars, strata = "caff", data = dat[dat$sex=="Male",],
                        test = F)

tab1 <- list(Female = tab1a, Male = tab1b)
print(tab1, showAllLevels = T, smd = T)
#> $Female
#>                        Stratified by caff
#>                         level                     No consumption <100 mg/day
#>   n                                                   203            916
#>   age (mean (SD))                                  59.33 (15.14)  62.92 (14.02)
#>   race (%)              Non-Hispanic White            38 (18.7)     286 (31.2)
#>                         Non-Hispanic Black            96 (47.3)     284 (31.0)
#>                         Mexican American              47 (23.2)     228 (24.9)
#>                         Other Hispanic                10 ( 4.9)      80 ( 8.7)
#>                         Other race                    12 ( 5.9)      38 ( 4.1)
#>   education (%)         Less than 9th grade           58 (28.6)     238 (26.0)
#>                         9-11th grade                  48 (23.6)     182 (19.9)
#>                         High school grade             43 (21.2)     224 (24.5)
#>                         Some college                  39 (19.2)     209 (22.8)
#>                         College graduate or above     15 ( 7.4)      63 ( 6.9)
#>   smoking (%)           Never smoker                 143 (70.4)     598 (65.3)
#>                         Current smoker                24 (11.8)     127 (13.9)
#>                         Former smoker                 36 (17.7)     191 (20.9)
#>   alcohol (%)           No consumption               192 (94.6)     826 (90.2)
#>                         <20 grams/day                  4 ( 2.0)      69 ( 7.5)
#>                         20+ grams/day                  7 ( 3.4)      21 ( 2.3)
#>   carbohyd (mean (SD))                             176.17 (73.88) 190.64 (73.34)
#>   physical.activity (%) Low                          95 (46.8)     393 (42.9)
#>                         Intermediate                 59 (29.1)     309 (33.7)
#>                         High                         49 (24.1)     214 (23.4)
#>   bmi.cat (%)           <20.0                          4 ( 2.0)       6 ( 0.7)
#>                         20.0 to <25.0                24 (11.8)     107 (11.7)
#>                         25.0 to <30.0                35 (17.2)     254 (27.7)
#>                         30.0 to <35.0                45 (22.2)     263 (28.7)
#>                         35.0 to <40.0                47 (23.2)     138 (15.1)
#>                         40.0+                        48 (23.6)     148 (16.2)
#>   htn (%)               No                           58 (28.6)     273 (29.8)
#>                         Yes                         145 (71.4)     643 (70.2)
#>   macrovascular (%)     No                          166 (81.8)     758 (82.8)
#>                         Yes                          37 (18.2)     158 (17.2)
```

```
#>   insulin (%)           No                          156 (76.8)     755 (82.4)
#>                         Yes                          47 (23.2)     161 (17.6)
#>   survey.cycle (%)      1999-00                      45 (22.2)     111 (12.1)
#>                         2001-02                      45 (22.2)      98 (10.7)
#>                         2003-04                      26 (12.8)     127 (13.9)
#>                         2005-06                      20 ( 9.9)     154 (16.8)
#>                         2007-08                      37 (18.2)     198 (21.6)
#>                         2009-10                      30 (14.8)     228 (24.9)
#>                        Stratified by caff
#>                         100-200 mg/day 200+ mg/day    SMD
#>   n                        424            332
#>   age (mean (SD))        62.04 (13.75)  60.19 (13.10)  0.150
#>   race (%)                154 (36.3)     189 (56.9)   0.527
#>                           108 (25.5)      40 (12.0)
#>                           111 (26.2)      69 (20.8)
#>                            30 ( 7.1)      20 ( 6.0)
#>                            21 ( 5.0)      14 ( 4.2)
#>   education (%)            79 (18.6)      56 (16.9)   0.203
#>                            89 (21.0)      76 (22.9)
#>                           109 (25.7)      81 (24.4)
#>                           109 (25.7)      89 (26.8)
#>                            38 ( 9.0)      30 ( 9.0)
#>   smoking (%)             238 (56.1)     136 (41.0)   0.377
#>                            81 (19.1)     119 (35.8)
#>                           105 (24.8)      77 (23.2)
#>   alcohol (%)             370 (87.3)     283 (85.2)   0.211
#>                            38 ( 9.0)      35 (10.5)
#>                            16 ( 3.8)      14 ( 4.2)
#>   carbohyd (mean (SD))  193.79 (72.18) 203.36 (76.05)  0.190
#>   physical.activity (%)   179 (42.2)     148 (44.6)   0.093
#>                           151 (35.6)     117 (35.2)
#>                            94 (22.2)      67 (20.2)
#>   bmi.cat (%)               7 ( 1.7)       4 ( 1.2)   0.250
#>                            40 ( 9.4)      44 (13.3)
#>                           109 (25.7)      80 (24.1)
#>                           133 (31.4)      82 (24.7)
#>                            73 (17.2)      63 (19.0)
#>                            62 (14.6)      59 (17.8)
#>   htn (%)                 123 (29.0)     110 (33.1)   0.052
#>                           301 (71.0)     222 (66.9)
```

```
#>   macrovascular (%)        352 (83.0)     266 (80.1)   0.042
#>                             72 (17.0)      66 (19.9)
#>   insulin (%)              340 (80.2)     252 (75.9)   0.094
#>                             84 (19.8)      80 (24.1)
#>   survey.cycle (%)          42 ( 9.9)      46 (13.9)   0.305
#>                             53 (12.5)      48 (14.5)
#>                             63 (14.9)      46 (13.9)
#>                             59 (13.9)      43 (13.0)
#>                            111 (26.2)      82 (24.7)
#>                             96 (22.6)      67 (20.2)
#>
#> $Male
#>                        Stratified by caff
#>                         level                      No consumption <100 mg/day
#>   n                                                   168            707
#>   age (mean (SD))                                  60.95 (12.68)  62.86 (12.88)
#>   race (%)              Non-Hispanic White            35 (20.8)     238 (33.7)
#>                         Non-Hispanic Black            78 (46.4)     209 (29.6)
#>                         Mexican American              37 (22.0)     190 (26.9)
#>                         Other Hispanic                13 ( 7.7)      48 ( 6.8)
#>                         Other race                     5 ( 3.0)      22 ( 3.1)
#>   education (%)         Less than 9th grade           50 (29.8)     194 (27.4)
#>                         9-11th grade                  41 (24.4)     135 (19.1)
#>                         High school grade             31 (18.5)     145 (20.5)
#>                         Some college                  30 (17.9)     136 (19.2)
#>                         College graduate or above     16 ( 9.5)      97 (13.7)
#>   smoking (%)           Never smoker                  72 (42.9)     281 (39.7)
#>                         Current smoker                34 (20.2)     155 (21.9)
#>                         Former smoker                 62 (36.9)     271 (38.3)
#>   alcohol (%)           No consumption               131 (78.0)     539 (76.2)
#>                         <20 grams/day                 16 ( 9.5)      96 (13.6)
#>                         20+ grams/day                 21 (12.5)      72 (10.2)
#>   carbohyd (mean (SD))                             195.01 (85.93) 213.00 (80.40)
#>   physical.activity (%) Low                          63 (37.5)     230 (32.5)
#>                         Intermediate                 54 (32.1)     284 (40.2)
#>                         High                         51 (30.4)     193 (27.3)
#>   bmi.cat (%)           <20.0                         5 ( 3.0)       4 ( 0.6)
#>                         20.0 to <25.0                26 (15.5)      95 (13.4)
#>                         25.0 to <30.0                53 (31.5)     276 (39.0)
#>                         30.0 to <35.0                46 (27.4)     185 (26.2)
```

```
#>                          35.0 to <40.0                 21 (12.5)      95 (13.4)
#>                          40.0+                         17 (10.1)      52 ( 7.4)
#>   htn (%)                No                            50 (29.8)     279 (39.5)
#>                          Yes                          118 (70.2)     428 (60.5)
#>   macrovascular (%)      No                           120 (71.4)     526 (74.4)
#>                          Yes                           48 (28.6)     181 (25.6)
#>   insulin (%)            No                           135 (80.4)     571 (80.8)
#>                          Yes                           33 (19.6)     136 (19.2)
#>   survey.cycle (%)       1999-00                       36 (21.4)      72 (10.2)
#>                          2001-02                       28 (16.7)      92 (13.0)
#>                          2003-04                       21 (12.5)     104 (14.7)
#>                          2005-06                       18 (10.7)     115 (16.3)
#>                          2007-08                       36 (21.4)     168 (23.8)
#>                          2009-10                       29 (17.3)     156 (22.1)
#>                         Stratified by caff
#>                          100-200 mg/day 200+ mg/day    SMD
#>   n                         461            569
#>   age (mean (SD))         61.67 (13.66)  61.70 (12.40)  0.074
#>   race (%)                  214 (46.4)     344 (60.5)   0.575
#>                              92 (20.0)      67 (11.8)
#>                              90 (19.5)     115 (20.2)
#>                              54 (11.7)      26 ( 4.6)
#>                              11 ( 2.4)      17 ( 3.0)
#>   education (%)              98 (21.3)      97 (17.0)   0.269
#>                              79 (17.1)      95 (16.7)
#>                             104 (22.6)     125 (22.0)
#>                             100 (21.7)     161 (28.3)
#>                              80 (17.4)      91 (16.0)
#>   smoking (%)               162 (35.1)     153 (26.9)   0.216
#>                             109 (23.6)     196 (34.4)
#>                             190 (41.2)     220 (38.7)
#>   alcohol (%)               349 (75.7)     427 (75.0)   0.080
#>                              63 (13.7)      80 (14.1)
#>                              49 (10.6)      62 (10.9)
#>   carbohyd (mean (SD))  225.58 (87.51) 237.26 (85.11)  0.273
#>   physical.activity (%)     153 (33.2)     195 (34.3)   0.099
#>                             177 (38.4)     199 (35.0)
#>                             131 (28.4)     175 (30.8)
#>   bmi.cat (%)                 4 ( 0.9)       4 ( 0.7)   0.181
#>                              61 (13.2)      80 (14.1)
```

```
#>                              169 (36.7)     187 (32.9)
#>                              132 (28.6)     146 (25.7)
#>                               64 (13.9)      95 (16.7)
#>                               31 ( 6.7)      57 (10.0)
#>   htn (%)                    190 (41.2)     234 (41.1)   0.126
#>                              271 (58.8)     335 (58.9)
#>   macrovascular (%)          353 (76.6)     412 (72.4)   0.066
#>                              108 (23.4)     157 (27.6)
#>   insulin (%)                375 (81.3)     449 (78.9)   0.032
#>                               86 (18.7)     120 (21.1)
#>   survey.cycle (%)            48 (10.4)      64 (11.2)   0.284
#>                               59 (12.8)      69 (12.1)
#>                               48 (10.4)     106 (18.6)
#>                               70 (15.2)      65 (11.4)
#>                              112 (24.3)     130 (22.8)
#>                              124 (26.9)     135 (23.7)
```

Survey design

The paper analyzed the data separately for males and females.
Let us create the survey design:

```
# Revised weight - weight divided by 6 cycles
dat.full$svy.weight <- dat.full$survey.weight/6
dat$svy.weight <- dat$survey.weight/6
summary(dat$svy.weight)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   223.2  1760.2  3349.3  4555.9  5817.8 25347.4

# Create an indicator variable
dat.full$miss <- 1
dat.full$miss[dat.full$id %in% dat$id] <- 0

# Set up the design
w.design0 <- svydesign(strata = ~strata, id = ~psu, weights = ~svy.weight,
                       data = dat.full, nest = TRUE)

# Subset the design
w.design1 <- subset(w.design0, miss == 0)
```

```r

# Subset the design for females
w.design.f <- subset(w.design1, sex == "Female")
dim(w.design.f)
#> [1] 1875   28

# Subset the design for males
w.design.m <- subset(w.design1, sex == "Male")
dim(w.design.m)
#> [1] 1905   28
```

Kaplan-Meier plot

Let us create the Kaplan-Meier plot for males and females:

```r
# KM for females
fit0.f <- svykm(Surv(stime, status) ~ caff, design = w.design.f)
plot(fit0.f)
```

```r

# KM for males
fit0.m <- svykm(Surv(stime, status) ~ caff, design = w.design.m)
plot(fit0.m)
```

Cox PH

Now, we will fit the Cox proportional hazards model, separately for males and females. Let us run the unadjusted model first.

```
# Unadjusted Cox PH for females
fit1.f <- svycoxph(Surv(stime, status) ~ caff, design = w.design.f)
publish(fit1.f)
#> Stratified 1 - level Cluster Sampling design (with replacement)
#> With (180) clusters.
#> subset(w.design1, sex == "Female")
#>  Variable          Units HazardRatio       CI.95   p-value
#>      caff No consumption         Ref
#>              <100 mg/day        0.80 [0.62;1.03]   0.08543
#>           100-200 mg/day        0.58 [0.42;0.80]   < 0.001
#>              200+ mg/day        0.61 [0.43;0.86]   0.00507

# Unadjusted Cox PH for males
fit1.m <- svycoxph(Surv(stime, status) ~ caff, design = w.design.m)
publish(fit1.m)
#> Stratified 1 - level Cluster Sampling design (with replacement)
#> With (180) clusters.
#> subset(w.design1, sex == "Male")
#>  Variable          Units HazardRatio       CI.95 p-value
#>      caff No consumption         Ref
```

```
#>              <100 mg/day        1.20 [0.88;1.63]   0.246
#>           100-200 mg/day        1.10 [0.75;1.59]   0.632
#>              200+ mg/day        1.10 [0.76;1.58]   0.607
```

Now, we will fit the Cox PH model, adjusting for covariates.

```
# Covariate adjusted Cox PH for females
fit2.f <- svycoxph(Surv(stime, status) ~ caff + age + race + education + smoking +
                     alcohol +  carbohyd + physical.activity + bmi.cat + htn +
                     macrovascular + insulin + survey.cycle, design = w.design.f)
publish(fit2.f)
#> Stratified 1 - level Cluster Sampling design (with replacement)
#> With (180) clusters.
#> subset(w.design1, sex == "Female")
#>           Variable                     Units HazardRatio       CI.95   p-value
#>               caff            No consumption         Ref
#>                                  <100 mg/day        0.62 [0.46;0.84]   0.00166
#>                               100-200 mg/day        0.44 [0.30;0.64]   < 0.001
#>                                  200+ mg/day        0.43 [0.28;0.66]   < 0.001
#>                age                                  1.08 [1.07;1.09]   < 0.001
#>               race        Non-Hispanic White         Ref
#>                           Non-Hispanic Black        0.81 [0.65;1.01]   0.06322
#>                             Mexican American        0.79 [0.61;1.03]   0.07976
#>                               Other Hispanic        0.65 [0.41;1.01]   0.05634
#>                                   Other race        0.64 [0.38;1.08]   0.09394
#>          education       Less than 9th grade         Ref
#>                                 9-11th grade        1.15 [0.84;1.57]   0.37396
#>                            High school grade        1.03 [0.75;1.41]   0.84759
#>                                 Some college        1.06 [0.79;1.44]   0.69484
#>                    College graduate or above        0.61 [0.40;0.93]   0.02192
#>            smoking              Never smoker         Ref
#>                               Current smoker        1.84 [1.42;2.38]   < 0.001
#>                                Former smoker        1.17 [0.95;1.46]   0.14379
#>            alcohol            No consumption         Ref
#>                                <20 grams/day        0.94 [0.65;1.36]   0.75056
#>                                20+ grams/day        1.05 [0.58;1.93]   0.86786
#>           carbohyd                                  1.00 [1.00;1.00]   0.69680
#>  physical.activity                        Low         Ref
#>                                 Intermediate        0.73 [0.57;0.92]   0.00834
#>                                         High        0.66 [0.49;0.89]   0.00638
```

```
#>            bmi.cat                      <20.0         Ref
#>                              20.0 to <25.0        0.39 [0.21;0.73]   0.00301
#>                              25.0 to <30.0        0.33 [0.19;0.57]   < 0.001
#>                              30.0 to <35.0        0.30 [0.17;0.54]   < 0.001
#>                              35.0 to <40.0        0.26 [0.13;0.49]   < 0.001
#>                                      40.0+        0.31 [0.16;0.62]   < 0.001
#>                htn                      No         Ref
#>                                        Yes        1.00 [0.80;1.24]   0.98084
#>      macrovascular                      No         Ref
#>                                        Yes        1.85 [1.48;2.32]   < 0.001
#>            insulin                      No         Ref
#>                                        Yes        1.87 [1.50;2.32]   < 0.001
#>       survey.cycle                  1999-00         Ref
#>                                     2001-02        0.58 [0.43;0.78]   < 0.001
#>                                     2003-04        0.75 [0.54;1.04]   0.08384
#>                                     2005-06        0.81 [0.57;1.15]   0.24147
#>                                     2007-08        0.88 [0.60;1.28]   0.49860
#>                                     2009-10        0.76 [0.56;1.04]   0.08292

# Covariate adjusted Cox PH for males
fit2.m <- svycoxph(Surv(stime, status) ~ caff + age + race + education + smoking +
                     alcohol +  carbohyd + physical.activity + bmi.cat + htn +
                     macrovascular + insulin + survey.cycle, design = w.design.m)
publish(fit2.m)
#> Stratified 1 - level Cluster Sampling design (with replacement)
#> With (180) clusters.
#> subset(w.design1, sex == "Male")
#>           Variable                      Units HazardRatio       CI.95   p-value
#>               caff            No consumption         Ref
#>                                  <100 mg/day        1.06 [0.75;1.49]   0.74199
#>                               100-200 mg/day        1.04 [0.69;1.57]   0.84938
#>                                  200+ mg/day        1.05 [0.71;1.54]   0.81249
#>                age                                  1.08 [1.06;1.09]   < 0.001
#>               race        Non-Hispanic White         Ref
#>                           Non-Hispanic Black        0.75 [0.61;0.91]   0.00423
#>                             Mexican American        0.68 [0.52;0.88]   0.00348
#>                               Other Hispanic        0.83 [0.51;1.36]   0.46367
#>                                   Other race        1.03 [0.60;1.76]   0.91073
#>          education       Less than 9th grade         Ref
#>                                 9-11th grade        1.47 [1.10;1.96]   0.00846
```

```
#>                            High school grade        1.03 [0.80;1.33]   0.81768
#>                                 Some college        1.02 [0.76;1.36]   0.91606
#>                     College graduate or above        0.68 [0.47;0.98]   0.03833
#>            smoking                Never smoker         Ref
#>                               Current smoker        1.90 [1.38;2.62]   < 0.001
#>                                Former smoker        1.03 [0.85;1.26]   0.74101
#>            alcohol             No consumption         Ref
#>                                <20 grams/day        0.95 [0.74;1.21]   0.67612
#>                                20+ grams/day        1.14 [0.83;1.58]   0.42116
#>           carbohyd                                  1.00 [1.00;1.00]   0.37461
#>  physical.activity                        Low         Ref
#>                                 Intermediate        0.64 [0.52;0.80]   < 0.001
#>                                         High        0.69 [0.56;0.87]   0.00137
#>            bmi.cat                     <20.0         Ref
#>                               20.0 to <25.0        0.25 [0.11;0.58]   0.00134
#>                               25.0 to <30.0        0.20 [0.09;0.48]   < 0.001
#>                               30.0 to <35.0        0.19 [0.08;0.45]   < 0.001
#>                               35.0 to <40.0        0.25 [0.10;0.64]   0.00380
#>                                       40.0+        0.26 [0.11;0.60]   0.00160
#>                htn                        No         Ref
#>                                          Yes        1.18 [0.98;1.42]   0.08442
#>      macrovascular                        No         Ref
#>                                          Yes        1.40 [1.17;1.66]   < 0.001
#>            insulin                        No         Ref
#>                                          Yes        1.48 [1.21;1.81]   < 0.001
#>       survey.cycle                   1999-00         Ref
#>                                      2001-02        1.37 [1.04;1.80]   0.02569
#>                                      2003-04        0.98 [0.68;1.43]   0.93245
#>                                      2005-06        1.07 [0.77;1.48]   0.67918
#>                                      2007-08        1.05 [0.78;1.41]   0.75981
#>                                      2009-10        0.83 [0.58;1.18]   0.29625
```

The adjusted results are approximately the same as in Table 2 of the article. Caffeine consumption was associated with mortality among women but not among men.

PH assumption

Now, we will check the proportional hazard assumption.

```r
# PH assumption among females
cox.zph(fit2.f)
#>                      chisq df    p
#> caff              3.65e-03  3 1.00
#> age               2.15e-06  1 1.00
#> race              2.87e-03  4 1.00
#> education         2.42e-03  4 1.00
#> smoking           2.44e-03  2 1.00
#> alcohol           1.14e-03  2 1.00
#> carbohyd          7.38e-04  1 0.98
#> physical.activity 3.34e-03  2 1.00
#> bmi.cat           2.21e-03  5 1.00
#> htn               1.33e-03  1 0.97
#> macrovascular     4.24e-04  1 0.98
#> insulin           2.26e-04  1 0.99
#> survey.cycle      3.49e-03  5 1.00
#> GLOBAL            2.55e-02 32 1.00

# PH assumption among males
cox.zph(fit2.m)
#>                      chisq df    p
#> caff              2.70e-03  3 1.00
#> age               2.45e-03  1 0.96
#> race              2.40e-03  4 1.00
#> education         1.13e-03  4 1.00
#> smoking           1.67e-03  2 1.00
#> alcohol           1.50e-03  2 1.00
#> carbohyd          1.33e-03  1 0.97
#> physical.activity 7.73e-03  2 1.00
#> bmi.cat           3.60e-03  5 1.00
#> htn               4.45e-06  1 1.00
#> macrovascular     4.32e-06  1 1.00
#> insulin           9.24e-05  1 0.99
#> survey.cycle      2.21e-03  5 1.00
#> GLOBAL            2.78e-02 32 1.00
```

The large p-values indicate that the proportional hazard assumption was met for both models.

Poisson

In this tutorial, we will see how to use Poisson and negative binomial regression. In practice, we use Poisson regression to model a count outcome. Note that we assume the mean is equal to the variance in Poisson. When the variance is greater than what's assumed by the model, overdispersion occurs. Poisson regression of overdispersed data leads to under-estimated or deflated standard errors, which leads to inflated test statistics and p-values. We can use negative binomial regression to model overdispersed data.

```r
# Load required packages
knitr::opts_chunk$set(echo = TRUE)
require(Publish)
require(survey)
```

Data

```r
# Load
load("Data/nonbinary/OAcvd.RData")

# Survey weight
analytic2$weight <- analytic2$weight/3

# Make fruit.cont as a numeric variable
analytic2$fruit.cont <- as.numeric(as.character(analytic2$fruit.cont))

# Make fruit.cont as a integer/count variable
analytic2$fruit.cont <- floor(analytic2$fruit.cont) # round

# Factor variables using lapply
var.names <- c("age", "sex", "income", "race", "bmicat", "phyact", "smoke",
```

```r
               "fruit", "painmed", "ht", "copd", "diab", "edu", "CVD", "OA")
analytic2[var.names] <- lapply(analytic2[var.names] , factor)
str(analytic2)
#> 'data.frame':    21623 obs. of  17 variables:
#>  $ CVD       : Factor w/ 2 levels "0 event","event": 1 1 1 1 1 1 1 1 1 1 ...
#>  $ age       : Factor w/ 4 levels "20-39 years",..: 2 3 4 1 3 1 3 1 3 3 ...
#>  $ sex       : Factor w/ 2 levels "Female","Male": 2 1 1 1 2 1 2 1 1 2 ...
#>  $ income    : Factor w/ 4 levels "$29,999 or less",..: 3 4 1 2 1 2 2 3 3 4 ...
#>  $ race      : Factor w/ 2 levels "Non-white","White": 2 2 2 2 2 2 2 2 2 2 ...
#>  $ bmicat    : Factor w/ 3 levels "Normal","Overweight",..: 1 1 2 2 2 1 2 1 2 2 ...
#>  $ phyact    : Factor w/ 3 levels "Active","Inactive",..: 3 1 1 2 1 3 2 3 2 3 ...
#>  $ smoke     : Factor w/ 3 levels "Current smoker",..: 2 3 3 2 2 2 2 2 3 2 ...
#>  $ fruit     : Factor w/ 3 levels "0-3 daily serving",..: 2 2 3 2 1 3 1 3 2 2 ...
#>  $ painmed   : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 1 1 ...
#>  $ ht        : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
#>  $ copd      : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
#>  $ diab      : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
#>  $ edu       : Factor w/ 4 levels "< 2ndary","2nd grad.",..: 2 2 1 4 4 4 1 4 4 1 ...
#>  $ weight    : num  4.8 13.29 4.43 11.1 9.61 ...
#>  $ OA        : Factor w/ 2 levels "Control","OA": 2 1 2 1 1 1 2 1 1 1 ...
#>  $ fruit.cont: num  3 4 8 3 2 10 1 8 5 3 ...
#>  - attr(*, "na.action")= 'omit' Named int [1:219757] 1 2 3 4 5 6 7 8 9 10 ...
#>   ..- attr(*, "names")= chr [1:219757] "3" "5" "7" "9" ...
```

Survey weighted summary

```r
# Survey design
w.design <- svydesign(id=~1, weights=~weight, data=analytic2)

# Cross-tabulation
xtabs(~fruit.cont, analytic2)
#> fruit.cont
#>    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
#>  407 1628 3304 4261 3759 2887 1980 1288  809  480  303  174  109   80   51   25
#>   16   17   18   19   20   21   22   23   24   25   26   27   29   42
#>   21   14    9    4    5    4    2    6    7    1    2    1    1    1
svytable(~fruit.cont, w.design)
#> fruit.cont
```

```
#>            0            1            2            3            4            5
#>   9085.00222  39974.10444  88216.88222 119206.45778 108665.00556  82149.23111
#>            6            7            8            9           10           11
#>  53791.70444  35463.49111  22593.73000  13023.83556   8878.24778   4520.85000
#>           12           13           14           15           16           17
#>   3476.42667   2210.76000   1911.70556    973.93444    571.37667    414.44444
#>           18           19           20           21           22           23
#>    270.82556     77.76222    253.13000     24.74778     34.17444    185.70778
#>           24           25           26           27           29           42
#>    154.67778    109.88444    149.81222     89.61444     14.91111     93.76889
svyhist(~fruit.cont, w.design)
```

```
svyby(~fruit.cont, ~phyact, w.design, svymean, deff = TRUE)
#>            phyact fruit.cont         se DEff.fruit.cont
#> Active     Active   5.123848 0.06310984        2.709145
#> Inactive Inactive   3.945868 0.03699451        2.394506
#> Moderate Moderate   4.468549 0.05297366        2.442263
```

Poisson regression

Let us fit the traditional (not design-adjusted) Poisson regression using the `glm` function:

```r
require(jtools)
analytic2$phyact2=relevel(analytic2$phyact, ref ="Inactive")

# Poisson regression - crude
fit1 <- glm(fruit.cont ~phyact2, data=analytic2, family=poisson)
summ(fit1, confint = TRUE, digits = 3)
```

Observations	21623
Dependent variable	fruit.cont
Type	Generalized linear model
Family	poisson
Link	log

$\chi^2(2)$	1219.347
Pseudo-R² (Cragg-Uhler)	0.055
Pseudo-R² (McFadden)	0.012
AIC	98570.197
BIC	98594.142

	Est.	2.5%	97.5%	z val.	p
(Intercept)	1.337	1.328	1.347	267.387	0.000
phyact2Active	0.274	0.259	0.289	34.983	0.000
phyact2Moderate	0.136	0.120	0.152	16.741	0.000

Standard errors: MLE

```r

# Poisson regression - adjusted for covariates
fit2 <- glm(fruit.cont ~ phyact2 + age + sex +
               income + race + bmicat +
               smoke + edu, data=analytic2, family=poisson)
summ(fit2, confint = TRUE, digits = 3, vifs = TRUE)
```

Survey weighted Poisson regression

Now, let's fit the design-adjusted Poisson regression using the `svyglm` function:

Observations	21623
Dependent variable	fruit.cont
Type	Generalized linear model
Family	poisson
Link	log

$\chi^2(17)$	2984.005
Pseudo-R² (Cragg-Uhler)	0.130
Pseudo-R² (McFadden)	0.030
AIC	96835.540
BIC	96979.207

	Est.	2.5%	97.5%	z val.	p	VIF
(Intercept)	1.156	1.123	1.189	68.773	0.000	NA
phyact2Active	0.250	0.235	0.266	31.432	0.000	1.040
phyact2Moderate	0.112	0.095	0.128	13.640	0.000	1.040
age40-49 years	0.027	0.010	0.043	3.196	0.001	1.102
age50-59 years	0.083	0.066	0.100	9.400	0.000	1.102
age60-64 years	0.127	0.103	0.151	10.287	0.000	1.102
sexMale	-0.173	-0.187	-0.160	-25.108	0.000	1.082
income$30,000-$49,999	0.039	0.017	0.060	3.557	0.000	1.144
income$50,000-$79,999	0.050	0.030	0.070	4.923	0.000	1.144
income$80,000 or more	0.083	0.062	0.103	7.964	0.000	1.144
raceWhite	-0.003	-0.024	0.018	-0.240	0.811	1.077
bmicatOverweight	-0.021	-0.035	-0.008	-3.054	0.002	1.096
bmicatUnderweight	-0.001	-0.034	0.033	-0.030	0.976	1.096
smokeFormer smoker	0.131	0.114	0.148	15.050	0.000	1.132
smokeNever smoker	0.181	0.163	0.199	19.436	0.000	1.132
edu2nd grad.	0.040	0.016	0.064	3.291	0.001	1.125
eduOther 2nd grad.	0.051	0.020	0.083	3.201	0.001	1.125
eduPost-2nd grad.	0.122	0.101	0.144	11.219	0.000	1.125

Standard errors: MLE

```r
require(jtools)
# Design
w.design <- update (w.design , phyact2=relevel(phyact, ref ="Inactive"))

```

```r
# Design-adjusted Poisson - crude
fit1 <- svyglm(fruit.cont ~phyact2, design=w.design, family=poisson)
summ(fit1, confint = TRUE, digits = 3)
```

Observations	21623
Dependent variable	fruit.cont
Type	Survey-weighted generalized linear model
Family	poisson
Link	log

Pseudo-R² (Cragg-Uhler)	0.064
Pseudo-R² (McFadden)	0.035
AIC	99087.436

	Est.	2.5%	97.5%	t val.	p
(Intercept)	1.373	1.354	1.391	146.410	0.000
phyact2Active	0.261	0.231	0.292	16.877	0.000
phyact2Moderate	0.124	0.095	0.154	8.230	0.000

Standard errors: Robust

```r

# Design-adjusted Poisson - adjusted for covariates
fit2<-svyglm(fruit.cont ~phyact2 + age + sex +
               income + race + bmicat +
               smoke + edu, design=w.design, family=poisson)
summ(fit2, confint = TRUE, digits = 3, vifs = TRUE)
```

Observations	21623
Dependent variable	fruit.cont
Type	Survey-weighted generalized linear model
Family	poisson
Link	log

Negative binomial regression

Let's fit the negative binomial regression model using the `glm` function. Below, we specify the dispersion parameter (theta) is

Pseudo-R² (Cragg-Uhler)	0.137
Pseudo-R² (McFadden)	0.076
AIC	97795.011

	Est.	2.5%	97.5%	t val.	p	VIF
(Intercept)	1.175	1.113	1.237	37.230	0.000	NA
phyact2Active	0.241	0.210	0.272	15.211	0.000	1.346
phyact2Moderate	0.105	0.074	0.135	6.775	0.000	1.346
age40-49 years	0.041	0.010	0.071	2.623	0.009	1.447
age50-59 years	0.102	0.071	0.133	6.402	0.000	1.447
age60-64 years	0.149	0.096	0.203	5.461	0.000	1.447
sexMale	-0.136	-0.161	-0.111	-10.500	0.000	1.148
income$30,000-$49,999	0.025	-0.015	0.066	1.217	0.224	1.294
income$50,000-$79,999	0.031	-0.008	0.070	1.559	0.119	1.294
income$80,000 or more	0.057	0.018	0.096	2.874	0.004	1.294
raceWhite	0.023	-0.012	0.059	1.289	0.197	1.193
bmicatOverweight	-0.009	-0.035	0.017	-0.708	0.479	1.331
bmicatUnderweight	0.029	-0.033	0.091	0.928	0.353	1.331
smokeFormer smoker	0.094	0.059	0.128	5.352	0.000	1.474
smokeNever smoker	0.140	0.102	0.177	7.290	0.000	1.474
edu2nd grad.	0.026	-0.020	0.072	1.093	0.274	1.302
eduOther 2nd grad.	0.033	-0.024	0.090	1.136	0.256	1.302
eduPost-2nd grad.	0.129	0.085	0.173	5.807	0.000	1.302

Standard errors: Robust

equal to 1, which suggests that the ratio of mean and variance is assumed to be 1.

```
require(MASS)
analytic2$phyact2=relevel(analytic2$phyact, ref ="Inactive")

# Negative binomial regression - crude
fit3<- glm(fruit.cont ~ phyact2, data=analytic2, family = negative.binomial(theta = 1))
round(exp(cbind(coef(fit3), confint(fit3))),2)
#> Waiting for profiling to be done...
#>                      2.5 % 97.5 %
#> (Intercept)     3.81  3.76   3.85
#> phyact2Active   1.32  1.29   1.34
```

```
#> phyact2Moderate 1.15  1.12   1.17

# Negative binomial regression - adjusted for covariates
fit4<-glm(fruit.cont ~phyact2 + age + sex + income + race + bmicat + smoke + edu, data=analytic
          family = negative.binomial(theta = 1))
round(exp(cbind(coef(fit4), confint(fit4))),2)
#> Waiting for profiling to be done...
#>                            2.5 % 97.5 %
#> (Intercept)           3.17  3.05   3.30
#> phyact2Active         1.29  1.26   1.31
#> phyact2Moderate       1.12  1.10   1.14
#> age40-49 years        1.03  1.01   1.05
#> age50-59 years        1.08  1.06   1.11
#> age60-64 years        1.14  1.10   1.17
#> sexMale               0.84  0.83   0.86
#> income$30,000-$49,999 1.05  1.02   1.07
#> income$50,000-$79,999 1.06  1.03   1.08
#> income$80,000 or more 1.09  1.07   1.12
#> raceWhite             0.99  0.97   1.02
#> bmicatOverweight      0.98  0.96   1.00
#> bmicatUnderweight     0.99  0.95   1.04
#> smokeFormer smoker    1.14  1.12   1.16
#> smokeNever smoker     1.20  1.17   1.23
#> edu2nd grad.          1.04  1.01   1.07
#> eduOther 2nd grad.    1.05  1.01   1.09
#> eduPost-2nd grad.     1.13  1.10   1.16
```

Survey weighted negative binomial regression

Now, let's fit the design-adjusted negative binomial regression model:

```
require(sjstats)

# Design-adjusted negative binomial - crude
fit3<- svyglm.nb(fruit.cont ~phyact2, design=w.design)
round(exp(cbind(coef(fit3), confint(fit3))),2)
#>                                2.5 %    97.5 %
#> theta.(Intercept)   28859.40 5797.98 143647.38
```

```
#> eta.(Intercept)         3.95    3.87      4.02
#> eta.phyact2Active       1.30    1.26      1.34
#> eta.phyact2Moderate     1.13    1.10      1.17

# Design-adjusted negative binomial - adjusted for covariates
fit4<-svyglm.nb(fruit.cont ~phyact2 + age + sex + income + race + bmicat + smoke + edu,
                design=w.design)
round(exp(cbind(coef(fit4), confint(fit4))),2)
#>                                        2.5 %     97.5 %
#> theta.(Intercept)         132340.69 15727.49 1113594.94
#> eta.(Intercept)                3.24     3.04       3.44
#> eta.phyact2Active              1.27     1.23       1.31
#> eta.phyact2Moderate            1.11     1.08       1.14
#> eta.age40-49 years             1.04     1.01       1.07
#> eta.age50-59 years             1.11     1.07       1.14
#> eta.age60-64 years             1.16     1.10       1.22
#> eta.sexMale                    0.87     0.85       0.90
#> eta.income$30,000-$49,999      1.03     0.99       1.07
#> eta.income$50,000-$79,999      1.03     0.99       1.07
#> eta.income$80,000 or more      1.06     1.02       1.10
#> eta.raceWhite                  1.02     0.99       1.06
#> eta.bmicatOverweight           0.99     0.97       1.02
#> eta.bmicatUnderweight          1.03     0.97       1.09
#> eta.smokeFormer smoker         1.10     1.06       1.14
#> eta.smokeNever smoker          1.15     1.11       1.19
#> eta.edu2nd grad.               1.03     0.98       1.07
#> eta.eduOther 2nd grad.         1.03     0.98       1.09
#> eta.eduPost-2nd grad.          1.14     1.09       1.19
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

R functions (N)

The list of new R functions introduced in this *non-binary data analysis* lab component are below:

Function_name	Package_name	Use
cox.zph	survival	To assess the proportional hazard assumption
coxph	survival	To fit cox regression model
ggforest	survminer	To produce a forest plot
multinom	nnet	To fit multinomial models
polr	MASS	To fir ordinal logistic and probit regressions
Surv	survival	To create a survival object
survdiff	survival	To compare survival times between groups
survfit	survival	To create survival curves
svy_vglm	svyVGAM	To fit design-based generalised linear models
svycoxph	survey	To fit cox regression model for complex survey data
svyglm.nb	sjstats	Negative binomial model for complex survey data
svykm	survey	Estimate survival function for complex survey data
svyolr	survey	Ordinal logistic for complex survey

Quiz (N)

> ⚠ Warning
>
> We are currently in the beta testing phase for our quiz page updates, so please be aware that some responses may not be accurate. Thank you for your patience.
> For those interested in downloading the quiz questions, you'll find a section below dedicated to downloading.

Live Quiz

You can click on the live quiz options to either see hints of why your response may be incorrect (response marked in red), or whether your choice of response is correct (response marked in green):

Download Quiz

1. **Downloading the File:**
   - Navigate to the link: See here.
   - Right-click on the link and select "Save link as…" from the dropdown menu.
   - Alternatively click here for downloading the quizzes.
   - Choose a destination folder on your computer where you'd like to save the file (e.g., Desktop). **Remember this location**, as you'll need to navigate to it later.
2. **Setting Up RStudio:**

- If you don't have RStudio installed, see the download link in here.
- Launch RStudio after installing.

3. **Installing Necessary R Packages:**

   - Before running the R Markdown file, ensure you have the required packages. In RStudio's console (located at the bottom left by default), enter the following commands to install them:

```r
install.packages("learnr")
install.packages("xfun")
```

4. **Opening and Running the File:**

   - In RStudio, go to `File` > `Open File...`.
   - Navigate to the folder where you saved the `Rmd` file and select it to open.
   - Once the file is open in RStudio, you'll see a "Run Document" button (green) at the top of the script editor. Click on it to run the R Markdown Quiz file.

Part XII

Longitudinal data

Background

The chapter focuses on longitudinal data analysis. The first dives into mixed effects models, highlighting their importance in studying repeated measurements, especially in longitudinal or clustered data. These models comprise two essential components: fixed effects, which represent universal trends across subjects or clusters, and random effects, capturing individual attributes of each subject or cluster. The tutorial explains their application using datasets, emphasizing model selection metrics and validation methods. The second tutorial introduces the generalized estimating equation (GEE), suitable for modeling non-normally distributed longitudinal data. GEE differentiates from mixed-effects models in its distribution assumptions and its capability to handle various correlation structures. The tutorial illustrates GEE's application, emphasizing its advantages over other regression models for repeated measurements, and ends with a comparison between GEE and mixed models' random effects.

> **!** Important
>
> **Datasets**:
> All of the datasets used in this tutorial can be accessed from this GitHub repository folder

Overview of tutorials

Mixed effects models

This tutorial provides an in-depth look into mixed effects models, which are instrumental in analyzing repeated measurements, especially in longitudinal or clustered data scenarios. Within the context of such models, there are two core components: fixed effects and random effects. Fixed effects depict broad trends applicable universally across subjects or clusters, offering insights such as average student performance across different subjects. In contrast, random effects spotlight the distinct attributes of each subject or cluster, accounting for

Longitudinal data analysis is specialized for handling data collected over time with repeated measurements on the same subjects. It accounts for within-subject correlation, time trends, and subject-specific effects, making it distinct from traditional regression, propensity score analysis, and machine learning approaches we discussed earlier, which are not designed for the longitudinal aspect of the data. Choosing the appropriate analysis method depends on the specific research question and the nature of the data at hand.

variables such as the quality of resources or the experience of teachers in schools. Using a dataset, the tutorial demonstrates the application and visualization of linear mixed effects models. Emphasis is also placed on model selection, using metrics like AIC and BIC, and on validating the assumptions of the model using residual plots and QQ-plots.

GEE

The tutorial introduces the generalized estimating equation (GEE) as a method for modeling longitudinal or clustered data that can be non-normally distributed, such as binary, count, or skewed data. Unlike mixed-effects models that assume a normal distribution for error terms and beta coefficients, GEE is distribution-free but assumes specific correlation structures within subjects or clusters. The tutorial exemplifies the application of GEE using respiratory data with a binary response variable. While logistic regression and Poisson regression handle binary and count data, they do not account for repeated measurement structures, potentially leading to incorrect standard error estimates. GEE allows for various correlation structures, and the tutorial details several types, including independent, exchangeable, AR-1 autoregressive, and unstructured correlations. The GEE models produce consistent estimates even if the correlation structure is misspecified. The tutorial concludes by comparing GEE with random effects in mixed models using several code examples.

> Tip
>
> **Optional Content**:
> You'll find that some sections conclude with an optional video walkthrough that demonstrates the code. Keep in mind that the content might have been updated since these videos were recorded. Watching these videos is optional.

> ⚠ Warning
>
> **Bug Report**:
> Fill out this form to report any issues with the tutorial.

Concepts (T)

Longitudinal Data Analysis

The section offers a brief overview of longitudinal data analysis, focusing on key concepts and methods. It explains that longitudinal data involves the repeated measurement of variables of interest over time, often referred to as panel data. The example study data is initially in wide-form, with multiple variables for each time point, and then it is reshaped into long-form data for analysis.

The analysis of longitudinal data includes various models, such as linear models, mixed-effect models, and marginal models (GEE). Linear models are introduced initially, where each subject has a common slope and intercept. These ideas can be expanded to incorporate random intercepts, random slopes, and combinations of both in mixed-effect models. Model diagnostics, including AIC, BIC, and -loglik, are discussed for model evaluation. The section briefly discussing different correlation structures and highlights the differences in interpretation between mixed models and marginal models. This section also includes an introduction to marginal structual model, which is a GEE model under a situation when treatment-confounder feedback exists.

Reading list

Key reference: (Faraway 2016; Hothorn and Everitt 2014)

Optional references: (M. E. Karim et al. 2021; Cui and Qian 2007)

Video Lessons

Longitudinal data formatting

Longitudinal models: mixed effect

Longitudinal models: GEE

Marginal structural model

Video Lesson Slides

Longitudinal models

Marginal structural model

Links

Longitudinal models

- Google Slides
- PDF Slides

Marginal structural model

- Google Slides
- PDF Slides
- GitHub page about marginal structural model simulation

References

Mixed effects models

In this section, we will learn about mixed effects models. Mixed effects models are popular choices for modeling repeated measurements, such as longitudinal or clustered data. Examples of longitudinal data include blood pressure measurements taken over time from the same individuals and CD4 count over time from the same individuals. Examples of clustered data include students within schools and patients within hospitals. There are two components in a mixed effects model: fixed effects and random effects:

- Fixed effects refer to general trends that are applicable to all subjects/clusters. This implies that we might want to investigate how students perform (on average) in different subjects such as math and history. These effects are commonly observed, i.e., fixed, across all schools.
- Random Effects capture the unique characteristics of each subject/cluster that differentiate them from one another. For instance, some schools may have better resources, more experienced teachers, or a more supportive learning environment. These differences are specific to each school, and we use random effects to capture them.

```r
# Load required packages
knitr::opts_chunk$set(echo = TRUE)
require(kableExtra)
require(Matrix)
require(jtools)
```

Repeated measures

In repeated measurement design, the response variable are measured multiple times for each individuals.

A useful reference on repeated measures is Section 9.2 of Faraway (2016).

Linear Mixed Effects Models for Repeated Measures Data

> **Important**
>
> - The linear mixed effect models are based on the idea that the correlation of an individual's responses depends on some unobserved individual characteristics.
> - In linear mixed effect models, we treat these unobserved characteristics as random effects in our model. If we could conditional on the random effects, then the repeated measurements can be assumed to be independent.

A useful reference on linear mixed effects models is Section 12.4 of Hothorn and Everitt (2014).

Data

We are going to use the `BtheB` dataset from the `HSAUR2` package to explain the linear mixed effect model:

```
library(HSAUR2)
data("BtheB")
kable(head(BtheB))%>%
  kable_styling(bootstrap_options = "striped", full_width = F,
                position = "left")
```

drug	length	treatment	bdi.pre	bdi.2m	bdi.3m	bdi.5m	bdi.8m
No	>6m	TAU	29	2	2	NA	NA
Yes	>6m	BtheB	32	16	24	17	20
Yes	<6m	TAU	25	20	NA	NA	NA
No	>6m	BtheB	21	17	16	10	9
Yes	>6m	BtheB	26	23	NA	NA	NA
Yes	<6m	BtheB	7	0	0	0	0

- A typical form of repeated measurement data from a clinical trial data.

- The individuals are allocated to different treatments then the response Beck Depression Inventory II were taken at baseline, 2, 3, 5, and 8 months

```r
## Box-plot of responses at different time points in treatment and control groups
data("BtheB", package = "HSAUR2")
layout(matrix(1:2, nrow = 1))
ylim <- range(BtheB[,grep("bdi", names(BtheB))],na.rm = TRUE)
tau <- subset(BtheB, treatment == "TAU")[, grep("bdi", names(BtheB))]
boxplot(tau, main = "Treated as Usual", ylab = "BDI",
        xlab = "Time (in months)",
        names = c(0, 2, 3, 5, 8),ylim = ylim)
btheb <- subset(BtheB, treatment == "BtheB")[, grep("bdi", names(BtheB))]
boxplot(btheb, main = "Beat the Blues", ylab = "BDI",
        xlab = "Time (in months)",
        names = c(0, 2, 3, 5, 8),ylim = ylim)
```

- The side-by-side box plots show the distributions of BDI overtime between control (Treated as Usual) and intervention (Beat the Blues) groups.
- As time goes, drops in BDI are more obvious in intervention which may indicate the intervention is effective.

Regular model fixed intercept and slope

To compare, we start with fixed effect linear model, i.e., a regular linear model without any random effect:

```r
## To analyze the data, we first need to convert the dataset to a analysis-ready form
BtheB$subject <- factor(rownames(BtheB))
nobs <- nrow(BtheB)
BtheB_long <- reshape(BtheB, idvar = "subject",
                      varying = c("bdi.2m", "bdi.3m", "bdi.5m",
                                  "bdi.8m"),
                      direction = "long")
BtheB_long$time <- rep(c(2, 3, 5, 8), rep(nobs, 4))
kable(head(BtheB_long))%>%
  kable_styling(bootstrap_options = "striped", full_width = F,
                position = "left")
```

	drug	length	treatment	bdi.pre	subject	time	bdi
1.2m	No	>6m	TAU	29	1	2	2
2.2m	Yes	>6m	BtheB	32	2	2	16
3.2m	Yes	<6m	TAU	25	3	2	20
4.2m	No	>6m	BtheB	21	4	2	17
5.2m	Yes	>6m	BtheB	26	5	2	23
6.2m	Yes	<6m	BtheB	7	6	2	0

```r
unique(BtheB_long$subject)
#>   [1] 1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18
#>  [19] 19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
#>  [37] 37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
#>  [55] 55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
#>  [73] 73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
#>  [91] 91  92  93  94  95  96  97  98  99  100
#> 100 Levels: 1 10 100 11 12 13 14 15 16 17 18 19 2 20 21 22 23 24 25 26 27 ... 99
kable(subset(BtheB_long,  subject == 2))%>%
  kable_styling(bootstrap_options = "striped", full_width = F,
                position = "left")
```

	drug	length	treatment	bdi.pre	subject	time	bdi
2.2m	Yes	>6m	BtheB	32	2	2	16
2.3m	Yes	>6m	BtheB	32	2	3	24
2.5m	Yes	>6m	BtheB	32	2	5	17
2.8m	Yes	>6m	BtheB	32	2	8	20

```
kable(subset(BtheB_long,  subject == 99))%>%
  kable_styling(bootstrap_options = "striped", full_width = F,
                position = "left")
```

	drug	length	treatment	bdi.pre	subject	time	bdi
99.2m	No	<6m	TAU	13	99	2	5
99.3m	No	<6m	TAU	13	99	3	5
99.5m	No	<6m	TAU	13	99	5	0
99.8m	No	<6m	TAU	13	99	8	6

```


lmfit <- lm(bdi ~ bdi.pre + time + treatment + drug +length,
            data = BtheB_long, na.action = na.omit)
require(jtools)
summ(lmfit)
```

Observations	280 (120 missing obs. deleted)
Dependent variable	bdi
Type	OLS linear regression

F(5,274)	35.78
R^2	0.40
Adj. R^2	0.38

Random intercept but fixed slope

Let us start with a model with a random intercept but fixed slope. In this case, the resulting regression line for each individual is parallel (for fixed slope) but have different intercepts (for random intercept).

	Est.	S.E.	t val.	p
(Intercept)	7.32	1.73	4.24	0.00
bdi.pre	0.57	0.05	10.44	0.00
time	-0.94	0.24	-3.97	0.00
treatmentBtheB	-3.32	1.10	-3.02	0.00
drugYes	-3.57	1.15	-3.11	0.00
length>6m	1.71	1.11	1.54	0.12

Standard errors: OLS

```r
## Fit a random intercept model with lme4 package
library("lme4")
BtheB_lmer1 <- lmer(bdi ~ bdi.pre + time + treatment + drug +length
                    + (1 | subject), data = BtheB_long,
                    REML = FALSE, na.action = na.omit)

summary(BtheB_lmer1)
#> Linear mixed model fit by maximum likelihood  ['lmerMod']
#> Formula: bdi ~ bdi.pre + time + treatment + drug + length + (1 | subject)
#>    Data: BtheB_long
#>
#>      AIC      BIC   logLik deviance df.resid
#>   1887.5   1916.6   -935.7   1871.5      272
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -2.6975 -0.5026 -0.0638  0.4124  3.8203
#>
#> Random effects:
#>  Groups   Name        Variance Std.Dev.
#>  subject  (Intercept) 48.78    6.984
#>  Residual             25.14    5.014
#> Number of obs: 280, groups:  subject, 97
#>
#> Fixed effects:
#>                Estimate Std. Error t value
#> (Intercept)     5.59239    2.24244   2.494
#> bdi.pre         0.63968    0.07789   8.212
#> time           -0.70476    0.14639  -4.814
```

```
#> treatmentBtheB -2.32908    1.67036  -1.394
#> drugYes        -2.82495    1.72684  -1.636
#> length>6m       0.19708    1.63832   0.120
#>
#> Correlation of Fixed Effects:
#>              (Intr) bdi.pr time   trtmBB drugYs
#> bdi.pre      -0.682
#> time         -0.238  0.020
#> tretmntBthB -0.390  0.121  0.018
#> drugYes      -0.073 -0.237 -0.022 -0.323
#> length>6m    -0.243 -0.242 -0.036  0.002  0.157
summ(BtheB_lmer1)
```

Observations	280
Dependent variable	bdi
Type	Mixed effects linear regression

AIC	1887.49
BIC	1916.57
Pseudo-R^2 (fixed effects)	0.39
Pseudo-R^2 (total)	0.79

Fixed Effects					
	Est.	S.E.	t val.	d.f.	p
(Intercept)	5.59	2.24	2.49	108.98	0.01
bdi.pre	0.64	0.08	8.21	104.08	0.00
time	-0.70	0.15	-4.81	199.32	0.00
treatmentBtheB	-2.33	1.67	-1.39	97.12	0.17
drugYes	-2.82	1.73	-1.64	98.20	0.11
length>6m	0.20	1.64	0.12	100.26	0.90

p values calculated using Satterthwaite d.f.

Important

- AIC, BIC, and -loglik etc are goodness-of-fit statistics, which tells you how well the model fits your

Random Effects		
Group	Parameter	Std. Dev.
subject	(Intercept)	6.98
Residual		5.01

Grouping Variables		
Group	# groups	ICC
subject	97	0.66

data. Since there is no standard to tell what values of these statistics are good, without comparison with other models, they have little information to tell.

- Fixed effects: this is the standard output we will have in any fixed-effect model. The interpretation of estimated coefficients will be similar to a regular linear model.
- You may compare the outputs with the regular linear model, then you will find that **lm** tends to **underestimate** the SE for estimated coefficients.

```
library(ggplot2)
BtheB_longna <- na.omit(BtheB_long)
dat <- data.frame(time=BtheB_longna$time,pred=fitted(BtheB_lmer1),
                  Subject= BtheB_longna$subject)
ggplot(data=dat,aes(x=time, y=pred, group=Subject,
                    color=Subject)) + theme_classic() +
    geom_line()
```

As we can see, for each individual, we have different intercepts but the slope over follow-up time is the same. Next, we will fit the model with random intercept and random slope.

Random intercept and random slope

In the codes below, we fitted a mixed effects model with both random intercept and random slope:

```
## We can fit a random slope and intercept model using lme4 package and treat variable time as
library("lme4")
BtheB_lmer2 <- lmer(bdi ~ bdi.pre + time + treatment + drug +length +
                    (time | subject), data = BtheB_long, REML = FALSE,
                    na.action = na.omit)

summ(BtheB_lmer2)
```

Observations	280
Dependent variable	bdi
Type	Mixed effects linear regression

The interpretation of the model outputs will be similar to the model with only random intercepts. Let us plot the data:

AIC	1891.04
BIC	1927.39
Pseudo-R^2 (fixed effects)	0.39
Pseudo-R^2 (total)	0.80

Fixed Effects					
	Est.	S.E.	t val.	d.f.	p
(Intercept)	5.61	2.25	2.50	106.79	0.01
bdi.pre	0.64	0.08	8.25	102.78	0.00
time	-0.70	0.15	-4.56	57.69	0.00
treatmentBtheB	-2.38	1.67	-1.42	97.12	0.16
drugYes	-2.87	1.73	-1.66	98.18	0.10
length>6m	0.14	1.64	0.09	100.04	0.93

p values calculated using Satterthwaite d.f.

Random Effects		
Group	Parameter	Std. Dev.
subject	(Intercept)	7.12
subject	time	0.43
Residual		4.90

Grouping Variables		
Group	# groups	ICC
subject	97	0.68

```r
library(ggplot2)
BtheB_longna <- na.omit(BtheB_long)
dat <- data.frame(time=BtheB_longna$time,pred=fitted(BtheB_lmer2),
                  Subject= BtheB_longna$subject)
ggplot(data=dat,aes(x=time, y=pred, group=Subject, color=Subject)) +
  theme_classic() +
    geom_line()
```

From the figure, we can see, we have different intercepts and different slopes over follow-up time for each individual.

Choice among models

A common question is to ask should I add random slope to our model or random intercept is good enough. We may want to compare the models in terms of AIC and BIC. Smaller values indicate a better model.

- **lm** usually will not be considered as a competitor of **lme** as they basically apply to different types of data.
- When choosing between random intercept and random slope, a quick solution is to fit all possible models then do likelihood ratio tests.
- For example, I am not sure whether I should use random intercept only or random intercept + random slope. I could fit both model, then do a likelihood ratio test:

```
anova(BtheB_lmer1, BtheB_lmer2)
#> Data: BtheB_long
#> Models:
#> BtheB_lmer1: bdi ~ bdi.pre + time + treatment + drug + length + (1 | subject)
#> BtheB_lmer2: bdi ~ bdi.pre + time + treatment + drug + length + (time | subject)
```

```
#>             npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
#> BtheB_lmer1    8 1887.5 1916.6 -935.75   1871.5
#> BtheB_lmer2   10 1891.0 1927.4 -935.52   1871.0 0.4542  2     0.7969
## The non-significant p-value shows that the second model is not
## statistically different from the first model. Therefore, adding a
## random slope is not necessary
```

The p-value is greater than 0.05, which indicate that adding a random slope does not make the fitting significantly better. To keep the model simple, we may just use random intercept.

Prediction of Random Effects

- Ref: (Hothorn and Everitt 2014) Section 12.5

If you have noticed in the R output of linear mixed effect model. Random effects are not estimated in the model.

- We could use the fitted model to predict random effects.
- Also, the predicted random effects can be used to examine the assumptions we have for linear mixed effect model.

The **ranef** function is used to predict the random effect in R

```
qint <- ranef(BtheB_lmer1)$subject[["(Intercept)"]]
qint
#>  [1] -16.36173411  -2.44276827  -3.81655251   3.05333928   3.45142248
#>  [6]   7.76598376   2.72263773  -7.22484569   6.84680617  -1.02552084
#> [11]   2.40417048   1.16611114  -8.55346533  -6.25844483  -2.26117896
#> [16]  -5.23031336   2.52509381  -1.64263857  -1.07523649   3.93689163
#> [21]   7.66903473 -16.38156595   1.74883174  -1.31871410  -9.02883854
#> [26]  -2.75150756   2.38884151   2.72740404  -3.41918542   6.27519245
#> [31]  -4.52154811  -8.67437225  -0.34470208  -0.63972054  -0.08927194
#> [36]   7.61914282   2.91050412  -2.58455318   4.24637616 -16.01664364
#> [41]   5.90993779   3.21014012  -7.04631398   3.09608824   4.71327710
#> [46]  16.70161338   2.26241522   2.26584476  15.77002952   1.75410838
#> [51]   6.18741474   1.99791717   0.56774362   4.04107968  10.72891321
#> [56]  -1.54551028  -5.28330207   2.04552526   2.36472056  -1.56879952
#> [61]   6.63565636   5.45929329  -4.19695096  -6.39632488  -2.21496288
```

```
#> [66]  -0.95956136  -3.96021851   7.17628718  -1.41644518  -4.99763144
#> [71]  -2.83374092  -1.28015494   8.79345988  -0.33213428   1.43146353
#> [76]  -3.13316295  -6.52989461   2.51445573   5.93415004   3.42003755
#> [81]   4.49954804   3.75178702  -6.44832897  12.88173218  -9.36110814
#> [86]   2.02028152 -18.09960241  -6.53428535   5.06613705  -3.31816912
#> [91]   7.26187445   0.03103101  -8.14350181  -4.89326763   2.92437698
#> [96]   5.24835927  -5.96778945
## predict random effects using the fitted model
```

Check assumptions

Remember, we have two assumptions in the linear mixed effect model with random intercept:

- error term follows normal distribution
- beta for subject i follows normal distribution

We have predict the values of random effect and we could extract residuals from the fitted model. Therefore, we can use QQ-plot to check their normality

```r
# Assumption 1
qres <- residuals(BtheB_lmer1)
qqnorm(qres, xlim = c(-3, 3), ylim = c(-20, 20),
       ylab = "Estimated residuals",
       main = "Residuals")
qqline(qres)
```

```r

# Assumption 2
qint <- ranef(BtheB_lmer1)$subject[["(Intercept)"]]
qqnorm(qint, ylab = "Estimated random intercepts",
       xlim = c(-3, 3), ylim = c(-20, 20),
       main = "Random intercepts")
qqline(qint)
```

Since points are almost on the lines, we can say that the normality assumption is met.

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Reference

GEE

In this section, we will learn about generalized estimating equation (GEE). GEE is another popular method for modeling longitudinal or clustered data.

> **Note**
>
> - GEE is a population-averaged (e.g., marginal) model, whereas mixed effects models are subject/cluster-specific. For example, we can use GEE when we are interested in exploring the population average effect. On the other hand, when we are interested in individual/cluster-specific effects, we can use the mixed effects models.
> - Also, in contrast to the mixed effects models where we assume error term and beta coefficients for subject i both follow normal distribution, GEE is distribution free. However, in GEE, we assume some correlation structures for within subjects/clusters. Hence, we can use GEE for non-normal data such as skewed data, binary data, or count data.

```r
# Load required packages
knitr::opts_chunk$set(echo = TRUE)
require(HSAUR2)
library(gee)
library(MuMIn)
library(geepack)
library("lme4")
require(afex)
```

Data Description

In addition to BtheB dataset in part 1, we used respiratory data from **HSAUR2** package to demonstrate the analysis with non-normal responses:

- The response variable in this dataset is **status** (the respiratory status), which is a binary response
- Other covariates are: treatment, age, gender, the study center
- The response has been measured at 0, 1, 2, 3, 4 mths for each subject

```r
data("respiratory", package = "HSAUR2")
head(respiratory)
#>     centre treatment gender age status month subject
#> 1        1   placebo female  46   poor     0       1
#> 112      1   placebo female  46   poor     1       1
#> 223      1   placebo female  46   poor     2       1
#> 334      1   placebo female  46   poor     3       1
#> 445      1   placebo female  46   poor     4       1
#> 2        1   placebo female  28   poor     0       2
```

Methods for Non-normal Distributions

- ref: (Hothorn and Everitt 2014) Section 13.2
- In addition to normally distributed response variables, the response variable can also follow non-normal distributions, such as binary or count responses.
- We have learned logistic regression or Poisson regression (glm) to analyze binary and count data but they are not considering the "repeated measurement" structure.
- The consequence of ignoring the longitudinal/repeated measurements structure is to have the wrong estimated standard errors for regression coefficients. Hence, our inference will be invalid. However, the glm still gives consistent estimated beta coefficients.
- There are many correlation structures in modelling GEE.

- With non-normal responses, different assumptions about these correlation structures can lead to different interpretations of the beta coefficients. Here, we will introduce two different GEE models: **marginal model** and **conditional model**.

Marginal Models

- ref: (Hothorn and Everitt 2014) Section 13.2.1

Repeated measurement and longitudinal data has responses taken at different time points. Therefore, we could simply review them as many series of cross-sectional data. Each cross-sectional data can be analyzed using glm introduced in previous lectures. Then a correlation structure can be assumed to connect different "cross-sections". The common correlation structures are:

An independent structure

An independent structure which assumes the repeated measures are independent. An example of independent structure is basically is an identity matrix:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

An exchangeable correlation

An exchangeable correlation assumes that for the each individual, the correlation between each pair of repeated measurements is the same, i.e., $Cor(Y_{ij}, Y_{ik}) = \rho$. In the correlation matrix, only ρ is unknown. Therefore, it is a single parameter working correlation matrix. An example of exchangeable correlation matrix is:

$$\begin{pmatrix} 1 & \rho & \rho \\ \rho & 1 & \rho \\ \rho & \rho & 1 \end{pmatrix}$$

An AR-1 autoregressive correlation

Defined as $Cor(Y_{ij}, Y_{ik}) = \rho^{|k-j|}$. If two measurements are taken at two closer time points the correlation is higher than these taken at two farther apart time points. It is also a single parameter working correlation matrix. An example of AR-1 correlation matrix is:

$$\begin{pmatrix} 1 & \rho & \rho^2 \\ \rho & 1 & \rho \\ \rho^2 & \rho & 1 \end{pmatrix}$$

An unstructured correlation

Different pairs of observation for each individual have different correlations $Cor(Y_{ij}, Y_{ik}) = \rho_{jk}$. Assume that each individual has K pairs of measurements, it is a $K(K-1)/2$ parameters working correlation matrix. An example of unstructured correlation matrix is:

$$\begin{pmatrix} 1 & \rho_{12} & \rho_{13} \\ \rho_{12} & 1 & \rho_{23} \\ \rho_{13} & \rho_{23} & 1 \end{pmatrix}$$

- Sometimes, specifying a "right" correlation matrix is hard. However, the marginal model (usually we use GEE) gives us consistent estimated coefficients even with misspecified correlation structure.

Conditional Models

- ref: (Hothorn and Everitt 2014) Section 13.2.2

- In GEE marginal models, the estimated regression coefficients are marginal (or population-averaged) effects. Therefore, the interpretation are at population-level. It is almost impossible to make inference on any specific individual or cluster.
- One solution is to do conditional models. The random effect approach in the part 1 can be extended to non-Gaussian response.

GEE models

After the short introduction of two models, let's take a look at real examples

- ref: (Hothorn and Everitt 2014) Section 13.3
- ref: (Faraway 2016) Section 10.2

Binary response

We started with binary response using respiratory dataset:

```
library(gee)
data("respiratory", package = "HSAUR2")
## Data manipulation
resp <- subset(respiratory, month > "0")
resp$baseline <- rep(subset(respiratory, month == "0")$status, rep(4, 111))
## Change the response to 0 and 1
resp$nstat <- as.numeric(resp$status == "good")
resp$month <- resp$month[, drop = TRUE]
```

Now we will fit a regular glm, i.e., model without random effect or any correlation structures. For binary outcomes, the estimated coefficients are log odds.

```
## Regular GLM
resp_glm <- glm(status ~ centre + treatment + gender + baseline+ age,
                data = resp, family = "binomial")
summary(resp_glm)
#>
```

```
#> Call:
#> glm(formula = status ~ centre + treatment + gender + baseline +
#>     age, family = "binomial", data = resp)
#>
#> Coefficients:
#>                     Estimate Std. Error z value Pr(>|z|)
#> (Intercept)        -0.900171   0.337653  -2.666  0.00768 **
#> centre2             0.671601   0.239567   2.803  0.00506 **
#> treatmenttreatment  1.299216   0.236841   5.486 4.12e-08 ***
#> gendermale          0.119244   0.294671   0.405  0.68572
#> baselinegood        1.882029   0.241290   7.800 6.20e-15 ***
#> age                -0.018166   0.008864  -2.049  0.04043 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 608.93  on 443  degrees of freedom
#> Residual deviance: 483.22  on 438  degrees of freedom
#> AIC: 495.22
#>
#> Number of Fisher Scoring iterations: 4
```

Now we will fit GEE with independent correlation structure:

```
## GEE with identity matrix
resp_gee.in <- gee(nstat ~ centre + treatment + gender + baseline + age,
                 data = resp, family = "binomial",
                 id = subject,corstr = "independence",
                 scale.fix = TRUE, scale.value = 1)
#> Beginning Cgee S-function, @(#) geeformula.q 4.13 98/01/27
#> running glm to get initial regression estimate
#>        (Intercept)            centre2 treatmenttreatment         gendermale
#>        -0.90017133         0.67160098         1.29921589         0.11924365
#>       baselinegood                age
#>         1.88202860        -0.01816588
summary(resp_gee.in)
#>
#>  GEE:  GENERALIZED LINEAR MODELS FOR DEPENDENT DATA
#>  gee S-function, version 4.13 modified 98/01/27 (1998)
```

```
#>
#> Model:
#>  Link:                      Logit
#>  Variance to Mean Relation: Binomial
#>  Correlation Structure:     Independent
#>
#> Call:
#> gee(formula = nstat ~ centre + treatment + gender + baseline +
#>     age, id = subject, data = resp, family = "binomial", corstr = "independence",
#>     scale.fix = TRUE, scale.value = 1)
#>
#> Summary of Residuals:
#>         Min          1Q      Median          3Q         Max
#> -0.93134415 -0.30623174  0.08973552  0.33018952  0.84307712
#>
#>
#> Coefficients:
#>                      Estimate  Naive S.E.   Naive z Robust S.E.   Robust z
#> (Intercept)        -0.90017133 0.337653052 -2.665965  0.46032700 -1.9555041
#> centre2             0.67160098 0.239566599  2.803400  0.35681913  1.8821889
#> treatmenttreatment  1.29921589 0.236841017  5.485603  0.35077797  3.7038127
#> gendermale          0.11924365 0.294671045  0.404667  0.44320235  0.2690501
#> baselinegood        1.88202860 0.241290221  7.799854  0.35005152  5.3764332
#> age                -0.01816588 0.008864403 -2.049306  0.01300426 -1.3969169
#>
#> Estimated Scale Parameter:  1
#> Number of Iterations:  1
#>
#> Working Correlation
#>      [,1] [,2] [,3] [,4]
#> [1,]    1    0    0    0
#> [2,]    0    1    0    0
#> [3,]    0    0    1    0
#> [4,]    0    0    0    1
```

- This model assumes an independent correlation structure, the output will be equal to glm.
- The outputs started from a summary of residuals
- The estimated coefficients are the same as GLM. For bi-

nary outcome, you may still interpret them as log odds. Naive SE and z value are estimated directly from this model. Robust SE and z are sandwich estimates.

- The difference between naive and robust indicates that the correlation structure may not be good.
- Working Correlation is the correlation structure estimated from the data (identity matrix for independence).

Let fit the GEE model with exchangeable correlation structure:

```
## GEE with exchangeable matrix
resp_gee.ex <- gee(nstat ~ centre + treatment + gender + baseline+ age,
                 data = resp, family = "binomial",
                 id = subject, corstr = "exchangeable",
                 scale.fix = TRUE, scale.value = 1)
#> Beginning Cgee S-function, @(#) geeformula.q 4.13 98/01/27
#> running glm to get initial regression estimate
#>        (Intercept)            centre2 treatmenttreatment         gendermale
#>        -0.90017133         0.67160098         1.29921589         0.11924365
#>       baselinegood                age
#>         1.88202860        -0.01816588
summary(resp_gee.ex)
#>
#>  GEE:  GENERALIZED LINEAR MODELS FOR DEPENDENT DATA
#>  gee S-function, version 4.13 modified 98/01/27 (1998)
#>
#> Model:
#>  Link:                      Logit
#>  Variance to Mean Relation: Binomial
#>  Correlation Structure:     Exchangeable
#>
#> Call:
#> gee(formula = nstat ~ centre + treatment + gender + baseline +
#>     age, id = subject, data = resp, family = "binomial", corstr = "exchangeable",
#>     scale.fix = TRUE, scale.value = 1)
#>
#> Summary of Residuals:
#>         Min          1Q      Median          3Q         Max
#> -0.93134415 -0.30623174  0.08973552  0.33018952  0.84307712
```

```
#>
#>
#> Coefficients:
#>                      Estimate Naive S.E.    Naive z Robust S.E.   Robust z
#> (Intercept)        -0.90017133  0.4784634 -1.8813796  0.46032700 -1.9555041
#> centre2             0.67160098  0.3394723  1.9783676  0.35681913  1.8821889
#> treatmenttreatment  1.29921589  0.3356101  3.8712064  0.35077797  3.7038127
#> gendermale          0.11924365  0.4175568  0.2855747  0.44320235  0.2690501
#> baselinegood        1.88202860  0.3419147  5.5043802  0.35005152  5.3764332
#> age                -0.01816588  0.0125611 -1.4462014  0.01300426 -1.3969169
#>
#> Estimated Scale Parameter:  1
#> Number of Iterations:  1
#>
#> Working Correlation
#>           [,1]      [,2]      [,3]      [,4]
#> [1,] 1.0000000 0.3359883 0.3359883 0.3359883
#> [2,] 0.3359883 1.0000000 0.3359883 0.3359883
#> [3,] 0.3359883 0.3359883 1.0000000 0.3359883
#> [4,] 0.3359883 0.3359883 0.3359883 1.0000000
```

- This model assumes an exchangeable correlation structure.
- The outputs started from a summary of residuals
- The estimated coefficients are the same GLM. For binary outcome, you may still interpret them as log odds. Naive S.E and z value are estimated directly from this model. Robust SE and z are sandwich estimates
- The difference between naive and robust is smaller, which indicates that the correlation structure is better specified.
- Working Correlation is the correlation structure estimated from the data

Let's check the estimated coefficients from all three models.

```
summary(resp_glm)$coefficients
#>                      Estimate  Std. Error    z value     Pr(>|z|)
#> (Intercept)        -0.90017133 0.337652992 -2.6659658 7.676750e-03
```

```
#> centre2             0.67160098 0.239566555  2.8034004 5.056684e-03
#> treatmenttreatment  1.29921589 0.236840962  5.4856047 4.120574e-08
#> gendermale          0.11924365 0.294671000  0.4046671 6.857223e-01
#> baselinegood        1.88202860 0.241290163  7.7998563 6.197770e-15
#> age                -0.01816588 0.008864401 -2.0493065 4.043215e-02
summary(resp_gee.in)$coefficients
#>                       Estimate  Naive S.E.   Naive z Robust S.E.   Robust z
#> (Intercept)        -0.90017133 0.337653052 -2.665965  0.46032700 -1.9555041
#> centre2             0.67160098 0.239566599  2.803400  0.35681913  1.8821889
#> treatmenttreatment  1.29921589 0.236841017  5.485603  0.35077797  3.7038127
#> gendermale          0.11924365 0.294671045  0.404667  0.44320235  0.2690501
#> baselinegood        1.88202860 0.241290221  7.799854  0.35005152  5.3764332
#> age                -0.01816588 0.008864403 -2.049306  0.01300426 -1.3969169
summary(resp_gee.ex)$coefficients
#>                       Estimate Naive S.E.    Naive z Robust S.E.   Robust z
#> (Intercept)        -0.90017133  0.4784634 -1.8813796  0.46032700 -1.9555041
#> centre2             0.67160098  0.3394723  1.9783676  0.35681913  1.8821889
#> treatmenttreatment  1.29921589  0.3356101  3.8712064  0.35077797  3.7038127
#> gendermale          0.11924365  0.4175568  0.2855747  0.44320235  0.2690501
#> baselinegood        1.88202860  0.3419147  5.5043802  0.35005152  5.3764332
#> age                -0.01816588  0.0125611 -1.4462014  0.01300426 -1.3969169
# Same estimated coefficients but different SEs
```

GEE with identity matrix is the same as GLM model. If we change the correlation structure to exchangeable does not change the beta estimates, but the naive SEs are closer to Robust SE, which indicates that the exchangeable correlation structure is a better reflection of the correlation structures.

Gaussian response

GEE can also be applied to Gaussian response

```
library(gee)
library(HSAUR2)
BtheB$subject <- factor(rownames(BtheB))
nobs <- nrow(BtheB)
BtheB_long <- reshape(BtheB, idvar = "subject",
                      varying = c("bdi.2m", "bdi.3m", "bdi.5m", "bdi.8m"),
```

```r
                      direction = "long")
BtheB_long$time <- rep(c(2, 3, 5, 8), rep(nobs, 4))
osub <- order(as.integer(BtheB_long$subject))
BtheB_long <- BtheB_long[osub,]
btb_gee.ind <- gee(bdi ~ bdi.pre + treatment + length + drug,
               data = BtheB_long, id = subject,
               family = gaussian, corstr = "independence")
#> Beginning Cgee S-function, @(#) geeformula.q 4.13 98/01/27
#> running glm to get initial regression estimate
#>    (Intercept)        bdi.pre treatmentBtheB      length>6m        drugYes
#>      3.5686314      0.5818494     -3.2372285      1.4577182     -3.7412982
summary(btb_gee.ind)
#>
#>  GEE:  GENERALIZED LINEAR MODELS FOR DEPENDENT DATA
#>  gee S-function, version 4.13 modified 98/01/27 (1998)
#>
#> Model:
#>  Link:                      Identity
#>  Variance to Mean Relation: Gaussian
#>  Correlation Structure:     Independent
#>
#> Call:
#> gee(formula = bdi ~ bdi.pre + treatment + length + drug, id = subject,
#>     data = BtheB_long, family = gaussian, corstr = "independence")
#>
#> Summary of Residuals:
#>         Min          1Q      Median          3Q         Max
#> -21.6497810  -5.8485100   0.1131663   5.5838383  28.1871039
#>
#>
#> Coefficients:
#>                  Estimate Naive S.E.   Naive z Robust S.E.   Robust z
#> (Intercept)     3.5686314  1.4833349  2.405816  2.26947617  1.5724472
#> bdi.pre         0.5818494  0.0563904 10.318235  0.09156455  6.3545274
#> treatmentBtheB -3.2372285  1.1295569 -2.865928  1.77459534 -1.8242066
#> length>6m       1.4577182  1.1380277  1.280916  1.48255866  0.9832449
#> drugYes        -3.7412982  1.1766321 -3.179667  1.78271179 -2.0986557
#>
#> Estimated Scale Parameter:  79.25813
#> Number of Iterations:  1
```

```
#>
#> Working Correlation
#>      [,1] [,2] [,3] [,4]
#> [1,]    1    0    0    0
#> [2,]    0    1    0    0
#> [3,]    0    0    1    0
#> [4,]    0    0    0    1
# require(Publish)
# publish(btb_gee.ind)
```

- This model assumes an independent correlation structure.
- The outputs started from a summary of residuals
- The estimated coefficient will be interpreted the same way as linear model. Naive S.E and z value are estimated directly from this model. Robust SE and z are sandwich estimates
- Working Correlation is the correlation struture estimated from the data (identity matrix for indepedence)

With exchangeable correlation matrix:

```
btb_gee.ex <- gee(bdi ~ bdi.pre + treatment + length + drug,
                data = BtheB_long, id = subject,
                family = gaussian, corstr = "exchangeable")
#> Beginning Cgee S-function, @(#) geeformula.q 4.13 98/01/27
#> running glm to get initial regression estimate
#>    (Intercept)        bdi.pre treatmentBtheB      length>6m        drugYes
#>      3.5686314      0.5818494     -3.2372285      1.4577182     -3.7412982
summary(btb_gee.ex)
#>
#>  GEE:  GENERALIZED LINEAR MODELS FOR DEPENDENT DATA
#>  gee S-function, version 4.13 modified 98/01/27 (1998)
#>
#> Model:
#>  Link:                      Identity
#>  Variance to Mean Relation: Gaussian
#>  Correlation Structure:     Exchangeable
#>
#> Call:
```

```
#> gee(formula = bdi ~ bdi.pre + treatment + length + drug, id = subject,
#>     data = BtheB_long, family = gaussian, corstr = "exchangeable")
#>
#> Summary of Residuals:
#>        Min         1Q     Median         3Q        Max
#> -23.955980  -6.643864  -1.109741   4.257688  25.452310
#>
#>
#> Coefficients:
#>                  Estimate Naive S.E.     Naive z Robust S.E.   Robust z
#> (Intercept)     3.0231602 2.30390185  1.31219140  2.23204410  1.3544357
#> bdi.pre         0.6479276 0.08228567  7.87412417  0.08351405  7.7583066
#> treatmentBtheB -2.1692863 1.76642861 -1.22806339  1.73614385 -1.2494854
#> length>6m      -0.1112910 1.73091679 -0.06429596  1.55092705 -0.0717577
#> drugYes        -2.9995608 1.82569913 -1.64296559  1.73155411 -1.7322940
#>
#> Estimated Scale Parameter:  81.7349
#> Number of Iterations:  5
#>
#> Working Correlation
#>           [,1]      [,2]      [,3]      [,4]
#> [1,] 1.0000000 0.6757951 0.6757951 0.6757951
#> [2,] 0.6757951 1.0000000 0.6757951 0.6757951
#> [3,] 0.6757951 0.6757951 1.0000000 0.6757951
#> [4,] 0.6757951 0.6757951 0.6757951 1.0000000
#publish(btb_gee.ex)
```

- The interpretation of estimated coefficients are similar to LM. Naive S.E and z value are estimated directly from this model. Robust SE and z are sandwich estimates
- Working Correlation is the correlation structure estimated from the data
- When we change the structure of correlation, the estimates and naive SE and z changed. A closer naive SE to robust SE indicates that the correlation structure is better specified.

Compare with Random Effects

- ref: (Hothorn and Everitt 2014) Section 13.4 We then use the conditional models (i.e., adding random effect) with non-normal response

Let us compare the GEE models with the mixed effect models.

```
## Generalized mixed effect model model
library("lme4")
resp_lmer <- glmer(nstat ~ baseline + month + treatment +
                   gender + age + centre +
                    (1 | subject), family = binomial(),
                  data = resp)
#> Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
#> Model failed to converge with max|grad| = 0.197453 (tol = 0.002, component 1)
```

```
require(afex)
resp_afex <- mixed(nstat ~ baseline + month + treatment +
                   gender + age + centre +
                    (1 | subject), family = binomial(),
                  data = resp, method = "LRT")
#> Contrasts set to contr.sum for the following variables: baseline, month, treatment, gender,
#> Numerical variables NOT centered on 0: age
#> If in interactions, interpretation of lower order (e.g., main) effects difficult.
#> Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
#> Model failed to converge with max|grad| = 0.00495318 (tol = 0.002, component 1)
#> Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
#> Model failed to converge with max|grad| = 0.0047889 (tol = 0.002, component 1)
#> Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
#> Model failed to converge with max|grad| = 0.0466122 (tol = 0.002, component 1)
#> Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
#> Model failed to converge with max|grad| = 0.0949357 (tol = 0.002, component 1)
#> Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
#> Model failed to converge with max|grad| = 0.041661 (tol = 0.002, component 1)
```

```
## GEE model
resp_gee3 <- gee(nstat ~ baseline + month + treatment +
                   gender + age + centre,
```

```r
                 data = resp, family = "binomial",
                 id = subject, corstr = "exchangeable",
                 scale.fix = TRUE, scale.value = 1)
#> Beginning Cgee S-function, @(#) geeformula.q 4.13 98/01/27
#> running glm to get initial regression estimate
#>        (Intercept)       baselinegood            month.L            month.Q
#>        -0.90363465         1.88945124        -0.14372490        -0.02455122
#>            month.C treatmenttreatment         gendermale                age
#>        -0.23255161         1.30410916         0.11969528        -0.01823703
#>            centre2
#>         0.67417628
library(MuMIn)
library(geepack)
resp_gee4 <- geeglm(nstat ~ baseline + month + treatment +
                   gender + age + centre,
                 data = resp, family = "binomial",
                 id = subject, corstr = "exchangeable",
                 scale.fix = TRUE)
resp_gee5 <- geeglm(nstat ~ baseline + month + treatment +
                   gender + age + centre,
                 data = resp, family = "binomial",
                 id = subject, corstr = "independence",
                 scale.fix = TRUE)
resp_gee6 <- geeglm(nstat ~ baseline + month + treatment +
                   gender + age + centre,
                 data = resp, family = "binomial",
                 id = subject, corstr = "ar1",
                 scale.fix = TRUE)
QIC(resp_gee4)
#>        QIC       QICu  Quasi Lik        CIC     params       QICC
#>  510.94098  499.71690 -240.85845   14.61204    9.00000  513.14098
QIC(resp_gee5)
#>        QIC       QICu  Quasi Lik        CIC     params       QICC
#>  511.14981  499.69791 -240.84895   14.72595    9.00000  512.93199
QIC(resp_gee6)
#>        QIC       QICu  Quasi Lik        CIC     params       QICC
#>  511.81242  500.27657 -241.13829   14.76792    9.00000  514.01242
# Smaller QIC values for correlation structure represents better models
# i.e., "exchangeable" in our case

```

```r
resp_gee4a <- geeglm(nstat ~ month + treatment + gender + age + centre,
                 data = resp, family = "binomial",
                 id = subject, corstr = "exchangeable",
                 scale.fix = TRUE)

resp_gee4b <- geeglm(nstat ~ treatment + gender + age + centre,
                 data = resp, family = "binomial",
                 id = subject, corstr = "exchangeable",
                 scale.fix = TRUE)

resp_gee4c <- geeglm(nstat ~ gender + age + centre,
                 data = resp, family = "binomial",
                 id = subject, corstr = "exchangeable",
                 scale.fix = TRUE)

resp_gee4d <- geeglm(nstat ~ age + centre,
                 data = resp, family = "binomial",
                 id = subject, corstr = "exchangeable",
                 scale.fix = TRUE)

resp_gee4e <- geeglm(nstat ~ centre,
                 data = resp, family = "binomial",
                 id = subject, corstr = "exchangeable",
                 scale.fix = TRUE)
QIC(resp_gee4)
#>        QIC       QICu  Quasi Lik        CIC     params       QICC
#>  510.94098  499.71690 -240.85845   14.61204    9.00000  513.14098
QIC(resp_gee4a)[2]
#>   QICu
#> 567.35
QIC(resp_gee4b)[2]
#>     QICu
#> 562.6247
QIC(resp_gee4c)[2]
#>     QICu
#> 585.3099
QIC(resp_gee4d)[2]
#>     QICu
#> 586.1581
QIC(resp_gee4e)[2]
```

```
#>     QICu
#> 592.3437
QIC(resp_gee4d)[2]
#>     QICu
#> 586.1581
# Covariates are selected based on the QICu criteria
```

```
## compare estimates (conditional vs. marginal)
summary(resp_lmer)$coefficients # Model failed to converge
#>                       Estimate Std. Error    z value     Pr(>|z|)
#> (Intercept)        -1.65464822 0.77621551 -2.1316866 3.303262e-02
#> baselinegood        3.08898642 0.59858787  5.1604561 2.463489e-07
#> month.L            -0.20348428 0.27957512 -0.7278340 4.667152e-01
#> month.Q            -0.02821472 0.27907476 -0.1011009 9.194703e-01
#> month.C            -0.35569321 0.28084908 -1.2664923 2.053369e-01
#> treatmenttreatment  2.16621998 0.55157850  3.9273104 8.590108e-05
#> gendermale          0.23836172 0.66606490  0.3578656 7.204439e-01
#> age                -0.02557199 0.01993989 -1.2824544 1.996833e-01
#> centre2             1.03849559 0.54182606  1.9166586 5.528132e-02
summary(resp_afex)$coefficients # Model failed to converge
#>                Estimate Std. Error    z value     Pr(>|z|)
#> (Intercept)  1.61366355 0.79597814  2.0272712 4.263469e-02
#> baseline1   -1.55321951 0.30467425 -5.0979678 3.433192e-07
#> month1       0.21660822 0.24441639  0.8862263 3.754956e-01
#> month2      -0.17701748 0.24250556 -0.7299523 4.654194e-01
#> month3       0.21559489 0.24440392  0.8821253 3.777090e-01
#> treatment1  -1.09162957 0.28098692 -3.8849836 1.023368e-04
#> gender1     -0.10313396 0.33932808 -0.3039358 7.611768e-01
#> age         -0.02576276 0.02031564 -1.2681244 2.047535e-01
#> centre1     -0.52829986 0.27639119 -1.9114207 5.595053e-02
summary(resp_gee3)$coefficients # Marginalized model
#>                       Estimate Naive S.E.    Naive z Robust S.E.   Robust z
#> (Intercept)        -0.90565507 0.47940039 -1.8891413  0.46042640 -1.9669920
#> baselinegood        1.86665412 0.34220231  5.4548261  0.35023043  5.3297885
#> month.L            -0.14330949 0.18044542 -0.7941986  0.18210027 -0.7869812
#> month.Q            -0.02448267 0.18034926 -0.1357514  0.18329974 -0.1335663
#> month.C            -0.23187407 0.18073662 -1.2829391  0.15929035 -1.4556693
#> treatmenttreatment  1.28311415 0.33592099  3.8196903  0.35055323  3.6602548
#> gendermale          0.11513183 0.41851611  0.2750953  0.44135628  0.2608592
#> age                -0.01785399 0.01258158 -1.4190585  0.01299686 -1.3737157
```

```
#> centre2              0.68481139 0.34010911  2.0135050  0.35681635  1.9192265
summary(resp_gee4)$coefficients # Marginalized model
#>                        Estimate    Std.err        Wald     Pr(>|W|)
#> (Intercept)        -0.90565577 0.46042559  3.86907740 4.918355e-02
#> baselinegood        1.86664028 0.35022955 28.40636763 9.834132e-08
#> month.L            -0.14330937 0.18209975  0.61934188 4.312920e-01
#> month.Q            -0.02448266 0.18329922  0.01784004 8.937453e-01
#> month.C            -0.23187385 0.15928992  2.11898053 1.454834e-01
#> treatmenttreatment  1.28310331 0.35055239 13.39730286 2.519864e-04
#> gendermale          0.11513002 0.44135474  0.06804584 7.942036e-01
#> age                -0.01785382 0.01299683  1.88706809 1.695330e-01
#> centre2             0.68481801 0.35681558  3.68351745 5.495281e-02
```

The significance of variables are similar in both variables, but the estimated coefficients are larger in generalized mixed effect model. However, it does not mean the estimated coefficients are inconsistent. Instead, two models are estimating different parameters. Remember, the mixed effect model is conditional on random effects, while the GEE is a marginalized model.

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Reference

R functions (T)

The list of new R functions introduced in this *longitudinal data analysis* lab component are below:

Function_name	Package_name	Use
gee	gee	To fit a generalized estimation equation model
geeglm	geepack	To fit a generalized estimation equation model
lmer	lme4	To fit linear mixed effects models
glmer	lme4	To fit generalized linear mixed effects models
mixed	afex	To fit generalized linear mixed effects models
qqnorm	base/stats	To fit a QQ plot
ranef	lme5	To extract the random effects from a model
reshape	base/stats	Reshape data, e.g., into wide to long or long to wide format
residuals	base/stats	To extract residuals of a model

Quiz (T)

> ⚠ Warning
>
> We are currently in the beta testing phase for our quiz page updates, so please be aware that some responses may not be accurate. Thank you for your patience.
> For those interested in downloading the quiz questions, you'll find a section below dedicated to downloading.

Live Quiz

You can click on the live quiz options to either see hints of why your response may be incorrect (response marked in red), or whether your choice of response is correct (response marked in green):

Download Quiz

1. **Downloading the File:**
   - Navigate to the link: See here.
   - Right-click on the link and select "Save link as…" from the dropdown menu.
   - Alternatively click here for downloading the quizzes.
   - Choose a destination folder on your computer where you'd like to save the file (e.g., Desktop). **Remember this location**, as you'll need to navigate to it later.
2. **Setting Up RStudio:**

- If you don't have RStudio installed, see the download link in here.
- Launch RStudio after installing.

3. **Installing Necessary R Packages:**

   - Before running the R Markdown file, ensure you have the required packages. In RStudio's console (located at the bottom left by default), enter the following commands to install them:

```r
install.packages("learnr")
install.packages("xfun")
```

4. **Opening and Running the File:**

   - In RStudio, go to `File` > `Open File...`.
   - Navigate to the folder where you saved the `Rmd` file and select it to open.
   - Once the file is open in RStudio, you'll see a "Run Document" button (green) at the top of the script editor. Click on it to run the R Markdown Quiz file.

Part XIII

Mediation analysis

Background

This chapter provides comprehensive tutorials on mediation analysis. The Baron and Kenny approach explores non-binary outcomes through directed acyclic graphs (DAGs) and regression in big data scenarios, with a focus on both continuous and binary mediators and outcomes. The justification of mediation analysis evaluates the connection between osteoarthritis (OA), pain medication, and cardiovascular disease (CVD), considering various covariates like BMI, smoking status, and associations with diseases like diabetes. The final mediation example centers on decomposing the total effect of OA on CVD through direct and indirect pathways via pain medication, including data preparation, weight computation, and outcome evaluation, accompanied by considerations of non-linearity and potential interactions.

> **!** Important
>
> **Datasets**:
> All of the datasets used in this tutorial can be accessed from this GitHub repository folder

Overview of tutorials

Baron and Kenny Approach

In the preceding chapters, we have primarily discussed about total effect of an exposure to the outcome. In this chapter, we will discuss about decomposition of the effect in the presence of a mediator.

The chapter, referencing the Baron and Kenny approach, delves into the analysis of non-binary outcomes using directed acyclic graphs (DAGs) and regression techniques for big data scenarios with a million observations. Initially, the chapter focuses on a continuous outcome and continuous mediator, where the true treatment effect is known. Through the data generating process, a DAG is formulated and data simulated, followed by an estimation of effects using generalized linear models. Subsequently, the Baron and Kenny approaches are applied to determine direct, total, and indirect effects. The chapter progresses to explore binary outcomes with both continuous and binary mediators, each time employing a similar approach: creating a

DAG, generating data, estimating effects using regression models, and then using the Baron and Kenny methodology to elucidate the relationships.

Justification of Mediation Analysis

In this chapter, the data analysis process centers on understanding the relationship between osteoarthritis (OA), pain medication, and cardiovascular disease (CVD) using a mediation analysis. Specifically, the analysis seeks to determine if OA, the exposure, is associated with an increased risk of CVD, the outcome. Additionally, it investigates whether pain medication acts as a mediator in this causal pathway. The total effect of OA on CVD risk was found to be significant. Furthermore, OA was observed to significantly influence the use of pain medication, which is the proposed mediator. To facilitate the analysis, the study considers various adjustment covariates, including demographic variables, confounders such as BMI and smoking status, and associations with other diseases like diabetes. The data used in this study is pre-processed, analyzed, and subsequently saved for further use.

Mediation Example

In the chapter, the focus is on decomposing the "total effect" of a given exposure, OA (A), on the outcome CVD (Y) into its natural direct effect (NDE; $A \to Y$) and a natural indirect effect (NIE) that routes through a mediator, in this case, pain medication (M). Initially, the required data is loaded and preprocessed. The mediation analysis involves several steps: (1) Preparing the data and ensuring it has the necessary variables; (2) Modifying data for different exposures and duplicating it; (3) Computing weights for the mediation based on logistic regressions, where the weights are applied to factor in the mediator's effect; (4) Building a weighted outcome model, which is a logistic regression to evaluate the outcome. To quantify these effects, the chapter derives point estimates for the total effect, direct effect, and indirect effect. Furthermore, confidence intervals for these effects are determined using bootstrap methods.

The results, including the proportion mediated by pain medication, are visualized using graphs. The chapter also delves into considerations of non-linearity and potential interactions between variables.

> Tip
>
> **Optional Content**:
> You'll find that some sections conclude with an optional video walkthrough that demonstrates the code. Keep in mind that the content might have been updated since these videos were recorded. Watching these videos is optional.

> Warning
>
> **Bug Report**:
> Fill out this form to report any issues with the tutorial.

Concepts (I)

Mediation Analysis

The section provides an overview of the concept and methods of mediation analysis in the context of epidemiology and statistical modeling. The section discusses total effects, indirect and direct paths, and highlights the limitations of the Baron and Kenny approach (Baron and Kenny 1986). It introduces the counterfactual definition, emphasizing the importance of adjusting for confounders and providing insights into the mechanics of mediation analysis through imputation and weighting methods. The section also covers sensitivity analysis, proportion mediated, and methodological extensions, including multi-category mediators and software references for conducting mediation analysis.

Reading list

Key reference: (Rochon, Bois, and Lange 2014; Lange et al. 2017)

Video Lessons

Mediation Analysis - Baron and Kenny (1986)

Setting up Mediation analysis with counterfactual definitions and why confounding adjustment helps

Mediation analysis mechanism under counterfactual definition (i) Outcome imputation & (ii) Weighting

Mediation analysis using survey data, assumptions, extensions, software and references

Video Lesson Slides

Links

- Google Slides
- PDF Slides

References

Baron and Kenny

Baron and Kenny (1986) approach 1

In Baron and Kenny approach 1 (Baron and Kenny 1986), two regression models are fitted to estimate the total effect, direct effect, and indirect effect.

$$Y = \alpha_0 + \beta_0 A + \gamma_0 M,$$
$$Y = \alpha_1 + \beta_1 A,$$

where Y is the outcome, A is the exposure, M is the mediator, and α, β, γ are regression coefficients. The effects are then calculated as:

- Total effect of A on Y: $\hat{\beta}_1$
- Direct effect of A on Y: $\hat{\beta}_0$
- Indirect effect of A on Y through M: $\hat{\beta}_1 - \hat{\beta}_0$,

where $\hat{\beta}$ is estimated regression coefficient of β.

Baron and Kenny (1986) approach 2

In the second approach, three models are fitted:

$$Y = \alpha_0 + \beta_0 A + \gamma_0 M,$$
$$Y = \alpha_1 + \beta_1 A,$$
$$M = \alpha_2 + \beta_2 A,$$

The indirect effect of A on Y through M can be calculated as: $\hat{\beta}_2 \times \hat{\beta}_0$, where $\hat{\beta}$ is estimated regression coefficient of β.

```r
# Load required packages
require(simcausal)
```

Big data: **What if we had 1,000,000 (1 million) observations?**

Let us explore the mediation analysis using Baron and Kenny (1986) approaches. We first simulate a big dataset and then show the results from the mediation analysis.

Continuous outcome, continuous mediator

- True treatment effect = 1.3

Data generating process

```r
require(simcausal)
D <- DAG.empty()
D <- D +
    node("A", distr = "rnorm", mean = 0, sd = 1) +
    node("M", distr = "rnorm", mean = 0.5 * A, sd = 1) +
    node("Y", distr = "rnorm", mean = 0.5 * M + 1.3 * A, sd = .1)
Dset <- set.DAG(D)
#> ...automatically assigning order attribute to some nodes...
#> node A, order:1
#> node M, order:2
#> node Y, order:3
```

Generate DAG

```r
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
#> using the following vertex attributes:
#> 120.8NAdarkbluenone0
#> using the following edge attributes:
#> 0.50.40.7black1
```

Generate Data

```
Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
#> simulating observed dataset from the DAG object
head(Obs.Data)
#>   ID           A          M          Y
#> 1  1 -0.56047565 -1.2883086 -1.3953599
#> 2  2 -0.23017749  1.2398506  0.3695715
#> 3  3  1.55870831  0.3103793  2.1781918
#> 4  4  0.07050839  1.5034478  0.8197663
#> 5  5  0.12928774  0.5072003  0.3199635
#> 6  6  1.71506499  1.0037356  2.6603537
```

Estimate effect (beta-coef)

```
fit <- glm(Y ~ A, family="gaussian", data=Obs.Data)
round(coef(fit),2)
#> (Intercept)           A
#>        0.00        1.55
fit.am <- glm(Y ~ A + M, family="gaussian", data=Obs.Data)
round(coef(fit.am),2)
#> (Intercept)           A           M
```

```
#>          0.0          1.3          0.5
fit.m <- glm(M ~ A, family="gaussian", data=Obs.Data)
round(coef(fit.m),2)
#> (Intercept)            A
#>          0.0          0.5
```

```
# from 1st model
a.coef <- round(coef(fit),2)[2]
a.coef
#>    A
#> 1.55
# from 2nd (adjusted) model
am.coef <- round(coef(fit.am),2)[2]
am.coef
#>   A
#> 1.3
m.coef <- round(coef(fit.am),2)[3]
m.coef
#>   M
#> 0.5
# from 3rd (mediator) model
ma.coef <- round(coef(fit.m),2)[2]
ma.coef
#>   A
#> 0.5
```

Baron and Kenny (1986) approach 1

```
# Direct effect
am.coef
#>   A
#> 1.3
# Total effect
a.coef
#>    A
#> 1.55
# Indirect effect
a.coef - am.coef
```

```
#>    A
#> 0.25
```

Baron and Kenny (1986) approach 2

```
# Indirect effect
m.coef * ma.coef
#>    M
#> 0.25
```

Binary outcome, continuous mediator

- True treatment effect = 1.3

Data generating process

```
require(simcausal)
D <- DAG.empty()
D <- D +
  node("A", distr = "rnorm", mean = 0, sd = 1) +
  node("M", distr = "rnorm", mean = 0.5 * A, sd = 1) +
  node("Y", distr = "rbern", prob = plogis(0.5 * M + 1.3 * A))
Dset <- set.DAG(D)
#> ...automatically assigning order attribute to some nodes...
#> node A, order:1
#> node M, order:2
#> node Y, order:3
```

Generate DAG

```
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
#> using the following vertex attributes:
```

```
#> 120.8NAdarkbluenone0
#> using the following edge attributes:
#> 0.50.40.7black1
```

Generate Data

```
Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
#> simulating observed dataset from the DAG object
head(Obs.Data)
#>   ID           A          M Y
#> 1  1 -0.56047565 -1.2883086 0
#> 2  2 -0.23017749  1.2398506 0
#> 3  3  1.55870831  0.3103793 1
#> 4  4  0.07050839  1.5034478 1
#> 5  5  0.12928774  0.5072003 1
#> 6  6  1.71506499  1.0037356 1
```

Estimate effect (beta-coef)

```
fit <- glm(Y ~ A, family=binomial(link = "logit"), data=Obs.Data)
round(coef(fit),2)
```

```
#> (Intercept)           A
#>        0.00        1.48
fit.am <- glm(Y ~ A + M, family=binomial(link = "logit"), data=Obs.Data)
round(coef(fit.am),2)
#> (Intercept)           A           M
#>         0.0         1.3         0.5
fit.m <- glm(M ~ A, family="gaussian", data=Obs.Data)
round(coef(fit.m),2)
#> (Intercept)           A
#>         0.0         0.5
```

```
# from 1st model
a.coef <- round(coef(fit),2)[2]
a.coef
#>    A
#> 1.48
# from 2nd (adjusted) model
am.coef <- round(coef(fit.am),2)[2]
am.coef
#>   A
#> 1.3
m.coef <- round(coef(fit.am),2)[3]
m.coef
#>   M
#> 0.5
# from 3rd (mediator) model
ma.coef <- round(coef(fit.m),2)[2]
ma.coef
#>   A
#> 0.5
```

Baron and Kenny (1986) approach 1

```
# Direct effect
am.coef
#>   A
#> 1.3
# Total effect
```

```r
a.coef
#>    A
#> 1.48
# Indirect effect
a.coef - am.coef
#>    A
#> 0.18
```

Baron and Kenny (1986) approach 2

```r
# Indirect effect
m.coef * ma.coef
#>    M
#> 0.25
```

Binary outcome, binary mediator

- True treatment effect = 1.3

Data generating process

```r
require(simcausal)
D <- DAG.empty()
D <- D +
  node("A", distr = "rnorm", mean = 0, sd = 1) +
  node("M", distr = "rbern", prob = plogis(0.5 * A)) +
  node("Y", distr = "rbern", prob = plogis(0.5 * M + 1.3 * A))
Dset <- set.DAG(D)
#> ...automatically assigning order attribute to some nodes...
#> node A, order:1
#> node M, order:2
#> node Y, order:3
```

Generate DAG

```
plotDAG(Dset, xjitter = 0.1, yjitter = .9,
        edge_attrs = list(width = 0.5, arrow.width = 0.4, arrow.size = 0.7),
        vertex_attrs = list(size = 12, label.cex = 0.8))
#> using the following vertex attributes:
#> 120.8NAdarkbluenone0
#> using the following edge attributes:
#> 0.50.40.7black1
```

Generate Data

```
Obs.Data <- sim(DAG = Dset, n = 1000000, rndseed = 123)
#> simulating observed dataset from the DAG object
head(Obs.Data)
#>   ID           A M Y
#> 1  1 -0.56047565 0 1
#> 2  2 -0.23017749 0 1
#> 3  3  1.55870831 0 1
#> 4  4  0.07050839 0 0
#> 5  5  0.12928774 1 0
#> 6  6  1.71506499 0 1
```

Estimate effect (beta-coef)

```
fit <- glm(Y ~ A, family=binomial(link = "logit"), data=Obs.Data)
round(coef(fit),2)
#> (Intercept)           A
#>        0.25        1.34
fit.am <- glm(Y ~ A + M, family=binomial(link = "logit"), data=Obs.Data)
round(coef(fit.am),2)
#> (Intercept)           A           M
#>         0.0         1.3         0.5
fit.m <- glm(M ~ A, family=binomial(link = "logit"), data=Obs.Data)
round(coef(fit.m),2)
#> (Intercept)           A
#>         0.0         0.5
```

```
# from 1st model
a.coef <- round(coef(fit),2)[2]
a.coef
#>    A
#> 1.34
# from 2nd (adjusted) model
am.coef <- round(coef(fit.am),2)[2]
am.coef
#>   A
#> 1.3
m.coef <- round(coef(fit.am),2)[3]
m.coef
#>   M
#> 0.5
# from 3rd (mediator) model
ma.coef <- round(coef(fit.m),2)[2]
ma.coef
#>   A
#> 0.5
```

Baron and Kenny (1986) approach 1

```r
# Direct effect
am.coef
#>   A
#> 1.3
# Total effect
a.coef
#>    A
#> 1.34
# Indirect effect
a.coef - am.coef
#>    A
#> 0.04
```

Baron and Kenny (1986) approach 2

```r
# Indirect effect
m.coef * ma.coef
#>    M
#> 0.25
```

As we can see, the estimated indirect effects are different from the two approaches. That means Baron and Kenny's (1986) approach doesn't work for a non-collapsible effect measure such as the odds ratio.

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Justification

We must show enough justification to do a mediation analysis. A causal diagram would be the first step to conceptualize the mediation analysis problem hypothetically. Then, we can empirically verify where doing the mediation analysis makes sense in that particular context.

```r
# Load required packages
require(survey)
require(DiagrammeR)
require(DiagrammeRsvg)
require(rsvg)
library(magrittr)
library(svglite)
library(png)
```

Pre-processing

Load saved data

```r
load("Data/mediation/cchs123pain.RData")
```

Prepared the data

```r
analytic.miss$mediator <- ifelse(analytic.miss$painmed == "Yes", 1, 0)
analytic.miss$exposure <- ifelse(analytic.miss$OA == "OA", 1, 0)
analytic.miss$outcome <- ifelse(analytic.miss$CVD == "event", 1, 0)
```

Notation

- Outcome (Y): Cardiovascular disease (CVD)
- Exposure (A): Osteoarthritis (OA)
- Mediator (M): Pain medication
- Adjustment covariates (C)

Hypothesis

1. For total effect (TE): Is OA (A) associated with CVD (Y)?

2. For mediation analysis: Does pain-medication (M) play a mediating role in the causal pathway between OA (A) and CVD (Y)? Here, we will decompose total effect (TE) to a natural direct effect (NDE) and a natural indirect effect (NIE).

Adjustment variables (C):

- Demographics
  - age
  - sex
  - income
  - race
  - education status
- Important confounders
  - BMI
  - physical activity
  - smoking status
  - fruit and vegetable consumption
- Relation with other diseases
  - hypertension
  - COPD
  - diabetes

Analysis/empirical exploration

Total effect

Outcome model (weighted) is

$logit[P(Y_a = 1|C = c] = \theta_0 + \theta_1 a + \theta_3 c$

Setting Design

```r
require(survey)
summary(analytic.miss$weight)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    0.39   21.76   42.21   66.70   81.07 2384.98

# Survey design
w.design0 <- svydesign(id=~1, weights=~weight, data=analytic.miss)
summary(weights(w.design0))
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    0.39   21.76   42.21   66.70   81.07 2384.98
sd(weights(w.design0))
#> [1] 80.34263

# Subset the design
w.design <- subset(w.design0, miss == 0)
summary(weights(w.design))
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#>    2.053   28.480   53.218   82.472  102.860 1295.970
sd(weights(w.design))
#> [1] 91.98946
```

```r
# Model
TE <- svyglm(outcome ~ exposure + age + sex + income + race + bmi + edu + phyact +
               smoke + fruit + diab, design = w.design, family = quasibinomial("logit"))
# painmed is mediator; not included here.
TE.save <- exp(c(summary(TE)$coef["exposure",1],
                confint(TE)["exposure",]))
TE.save
#>             2.5 %   97.5 %
#> 1.537005 1.230735 1.919489
```

Exposure to OA has a detrimental effect on CVD risk (significant!).

Effect on the mediators

```r
fit.m = svyglm(mediator ~ exposure + age + sex + income + race + bmi + edu +
                 phyact + smoke + fruit + diab, design = w.design,
               family = binomial("logit"))
#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
med.save <- exp(c(summary(fit.m)$coef["exposure",1], confint(fit.m)["exposure",]))
med.save
#>              2.5 %   97.5 %
#> 2.428463 2.059265 2.863853
```

Exposure to OA has a substantial effect on the mediator (Pain medication) as well (significant!). Hence, it would be interesting to explore a mediation analysis to assess the mediating role.

Save data

```r
save(w.design, file = "Data/mediation/cchs123painW.RData")
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Mediation Example

```r
# Load required packages
knitr::opts_chunk$set(echo = TRUE)
require(survey)
require(DiagrammeR)
require(DiagrammeRsvg)
require(rsvg)
library(magrittr)
library(svglite)
library(png)
require(Publish)
```

We want to decompose of the "total effect" of a given exposure OA (A) on the outcome CVD (Y) into

- a natural direct effect (NDE; $A \to Y$) and
- a natural indirect effect (NIE) through a mediator pain medication (M) through ($A \to M \to Y$).

Step 0: Build data first

```r
load("Data/mediation/cchs123pain.RData")
source("Data/mediation/medFunc.R")
ls()
#> [1] "analytic.cc"        "analytic.miss"      "doEffectDecomp"
#> [4] "doEffectDecomp.int" "has_annotations"

varlist <- c("age", "sex", "income", "race", "bmi", "edu", "phyact", "smoke", "fruit", "diab")
analytic.miss$mediator <- ifelse(analytic.miss$painmed == "Yes", 1, 0)
analytic.miss$exposure <- ifelse(analytic.miss$OA == "OA", 1, 0)
analytic.miss$outcome <- ifelse(analytic.miss$CVD == "event", 1, 0)
```

Pre-run step 3 model

We will utilize this fit in step 3

```
# A = actual exposure (without any change)
analytic.miss$exposureTemp <- analytic.miss$exposure

# Design
w.design0 <- svydesign(id=~1, weights=~weight, data=analytic.miss)
w.design <- subset(w.design0, miss == 0)

# Replace exposure with exposureTemp. This will be necessary in step 3
fit.m <- svyglm(mediator ~ exposureTemp +
                 age + sex + income + race + bmi + edu + phyact + smoke + fruit + diab,
                design = w.design, family = binomial("logit"))
#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
publish(fit.m)
#>      Variable             Units OddsRatio       CI.95     p-value
#>  exposureTemp                        2.43 [2.06;2.86]     < 1e-04
#>           age       20-29 years       Ref
#>                     30-39 years      1.00 [0.88;1.13]   0.9442989
#>                     40-49 years      0.93 [0.82;1.06]   0.2651302
#>                     50-59 years      0.66 [0.58;0.76]     < 1e-04
#>                     60-64 years      0.61 [0.51;0.72]     < 1e-04
#>               65 years and over      0.61 [0.52;0.71]     < 1e-04
#>           sex            Female       Ref
#>                             Male      0.50 [0.46;0.55]     < 1e-04
#>        income   $29,999 or less       Ref
#>                 $30,000-$49,999      1.20 [1.06;1.35]   0.0043533
#>                 $50,000-$79,999      1.21 [1.08;1.37]   0.0014914
#>                 $80,000 or more      1.28 [1.14;1.45]     < 1e-04
#>          race         Non-white       Ref
#>                            White      1.81 [1.62;2.02]     < 1e-04
#>           bmi       Underweight       Ref
#>                  healthy weight      1.09 [0.82;1.44]   0.5631582
#>                      Overweight      1.33 [1.01;1.77]   0.0449616
#>           edu           < 2ndary       Ref
#>                       2nd grad.      1.13 [0.98;1.30]   0.1014986
#>                 Other 2nd grad.      1.30 [1.08;1.55]   0.0050596
#>                  Post-2nd grad.      1.25 [1.10;1.42]   0.0008252
#>        phyact            Active       Ref
```

```
#>                        Inactive      1.12 [1.02;1.23]   0.0184447
#>                        Moderate      1.12 [1.01;1.25]   0.0364592
#>          smoke      Never smoker       Ref
#>                  Current smoker      1.29 [1.16;1.44]     < 1e-04
#>                   Former smoker      1.28 [1.17;1.40]     < 1e-04
#>         fruit 0-3 daily serving       Ref
#>               4-6 daily serving      0.92 [0.83;1.02]   0.0976967
#>                6+ daily serving      0.80 [0.71;0.90]   0.0001979
#>          diab                 No       Ref
#>                              Yes      1.23 [0.99;1.52]   0.0626501
```

Step 1 and 2: Replicate data with different exposures

We manipulate and duplicate data here

```r
dim(analytic.miss)
#> [1] 397173     28
dim(analytic.cc)
#> [1] 28734    23
nrow(analytic.miss) - nrow(analytic.cc)
#> [1] 368439

# Create counterfactual data. This will be necessary in step 3
d1 <- d2 <- analytic.miss

# Exposed = Exposed
d1$exposure.counterfactual <- d1$exposure

# Exposed = Not exposed
d2$exposure.counterfactual <- !d2$exposure

# duplicated data (double the amount)
newd <- rbind(d1, d2)
newd <- newd[order(newd$ID), ]
dim(newd)
#> [1] 794346     29
```

Step 3: Compute weights for the mediation

Weight is computed by

$W^{M|C} = \frac{P(M|A^*,C)}{P(M|A,C)}$

in all data `newd` (fact `d1` + alternative fact `d2`).

- $P(M|A,C)$ is computed from a logistic regression of M on $A + C$.
  - $logit[P(M = 1|C = c]) = \beta_0 + \beta_1 a + \beta_3 c$
- $P(M|A^*,C)$ is computed from a logistic regression of M on $A^* + C$.
  - $logit[P(M = 1|C = c]) = \beta_0 + \beta_1' a^* + \beta_3' c$

```r
# First, use original exposure (all A + all A):
# A = actual exposure (without any change)
# A = exposure
newd$exposureTemp <- newd$exposure

# Probability of M given A + C
w <- predict(fit.m, newdata=newd, type='response')
direct <- ifelse(newd$mediator, w, 1-w)

# Second, use counterfactual exposures (all A + all !A):
# A* = Opposite (counterfactual) values of the exposure
# A* = exposure.counterfactual
newd$exposureTemp <- newd$exposure.counterfactual

# Probability of M given A* + C
w <- predict(fit.m, newdata=newd, type='response')
indirect <- ifelse(newd$mediator, w, 1-w)

# Mediator weights
newd$W.mediator <- indirect/direct
summary(newd$W.mediator)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>     0.4     1.0     1.0     1.0     1.2     2.2  670758
hist(newd$W.mediator)
```

Incorporating the survey weights:

Note: scaling can often be helpful if there exists extreme weights.

```
# scale survey weights
#newd$S.w <- with(newd,(weight)/mean(weight))
newd$S.w <- with(newd,weight)
newd$S.w[is.na(newd$S.w)]
#> numeric(0)
summary(newd$S.w)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    0.39   21.76   42.21   66.70   81.07 2384.98

# Multiply mediator weights with scaled survey weights
newd$SM.w <- with(newd,(W.mediator * S.w))

summary(newd$SM.w)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>     0.4    24.7    46.4    73.9    88.8  3531.6  670758
table(newd$miss[is.na(newd$SM.w)])
#>
#>      1
#> 670758
newd$SM.w[is.na(newd$SM.w)] <- 0
summary(newd$SM.w)
```

```
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>     0.0     0.0     0.0    11.5     0.0  3531.6

hist(newd$SM.w, main = "", xlab = "Combined weights",
     ylab = "Frequency", freq = TRUE)
```

Here all missing weights are associated with incomplete cases (`miss==1`)! Hence, doesn't matter if they are missing or other value (0) in them.

Step 4: Weighted outcome Model

Outcome model is

$logit[P(Y_{a,M(a^*)} = 1|C = c)] = \theta_0 + \theta_1 a + \theta_2 a^* + \theta_3 c$

after weighting (combination of mediator weight + sampling weight).

```
# Outcome analysis
w.design0 <- svydesign(id=~1, weights=~SM.w, data=newd)
w.design <- subset(w.design0, miss == 0)

# Fit Y on (A + A* + C)
```

```r
fit <- svyglm(outcome ~ exposure + exposure.counterfactual +
                age + sex + income + race + bmi + edu + phyact + smoke + fruit + diab,
             design = w.design, family = binomial("logit"))
#> Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

Point estimates

Following are the conditional ORs:

- $OR_{TE}(C = c) = \exp(\theta_1 + \theta_2)$
- $OR_{NDE}(A = 1, M = 0, C = c) = \exp(\theta_1)$
- $OR_{NIE}(A^* = 1, M = 0, C = c) = \exp(\theta_2)$

```r
# total effect of A-> Y + A -> M -> Y
TE <- exp(sum(coef(fit)[c('exposure', 'exposure.counterfactual')]))
TE
#> [1] 1.544694

# direct effect of A-> Y (not through M)
DE <- exp(unname(coef(fit)['exposure']))
DE
#> [1] 1.488554

# indirect effect of A-> Y (A -> M -> Y)
IE <- exp(unname(coef(fit)[c('exposure.counterfactual')]))
IE
#> [1] 1.037714

# Product of ORs; same as TE
DE * IE
#> [1] 1.544694

# Proportion mediated
PM <- log(IE) / log(TE)
PM
#> [1] 0.08513902
```

Obtaining results fast

User-written function `doEffectDecomp()` (specific to OA-CVD problem):

```
doEffectDecomp(analytic.miss, ind = NULL, varlist = varlist)
#>          TE         DE         IE         PM
#> 1.54469393 1.48855395 1.03771444 0.08513902
# function is provided in the appendix
```

Confidence intervals

Standard errors and confidence intervals are determined by bootstrap methods.

```
require(boot)
#> Loading required package: boot
#>
#> Attaching package: 'boot'
#> The following object is masked from 'package:survival':
#>
#>     aml
# I ran the computation on a 24 core computer,
# hence set ncpus = 5 (keep some free).
# If you have more / less cores, adjust accordingly.
# Try parallel package to find how many cores you have.
# library(parallel)
# detectCores()
# doEffectDecomp is a user-written function
# See appendix for the function
set.seed(504)
bootresBin <- boot(data=analytic.miss, statistic=doEffectDecomp,
                R = 5, parallel = "multicore", ncpus=5,
                varlist = varlist)
```

R = 5 is not reliable for bootstrap. In real applications, try 250 at least.

```r
bootci1b <- boot.ci(bootresBin,type = "perc",index=1)
#> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints
bootci2b <- boot.ci(bootresBin,type = "perc",index=2)
#> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints
bootci3b <- boot.ci(bootresBin,type = "perc",index=3)
#> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints
bootci4b <- boot.ci(bootresBin,type = "perc",index=4)
#> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints
```

```r
# Number of bootstraps
bootresBin$R
#> [1] 5

# Total Effect
c(bootresBin$t0[1], bootci1b$percent[4:5])
#>       TE
#> 1.544694 1.293208 1.894417

# Direct Effect
c(bootresBin$t0[2], bootci2b$percent[4:5])
#>       DE
#> 1.488554 1.303554 1.876916

# Indirect Effect
c(bootresBin$t0[3], bootci3b$percent[4:5])
#>        IE
#> 1.0377144 0.9738072 1.0093246

# Proportion Mediated
c(bootresBin$t0[4], bootci4b$percent[4:5])
#>          PM
#>  0.08513902 -0.08360848  0.01655013
```

The proportion mediated through pain medication was about 8.51% on the log odds ratio scale.

Visualization for main effects

```r
require(plotrix)
TEc <- c(bootresBin$t0[1], bootci1b$percent[4:5])
DEc <- c(bootresBin$t0[2], bootci2b$percent[4:5])
IEc <- c(bootresBin$t0[3], bootci3b$percent[4:5])
mat <- rbind(TEc,DEc,IEc)
colnames(mat) <- c("Point", "2.5%", "97.5%")
mat
#>        Point      2.5%    97.5%
#> TEc 1.544694 1.2932078 1.894417
#> DEc 1.488554 1.3035540 1.876916
#> IEc 1.037714 0.9738072 1.009325

plotCI(1:3, mat[,1], ui=mat[,3], li=mat[,2], xlab = "Estimates", ylab = "", xaxt="n")
axis(1, at=1:3,labels=c("TE","NDE","NIE"))
abline(h=1, lty = 2)
```

Non-linearity

Consider

- non-linear relationships (polynomials) and interactions between exposure, demographic/baseline covariates and mediators,
- Is misclassification of the mediators possible?

Here we are again using a user-written function `doEffectDecomp.int()` (including interaction `phyact*diab` in the mediation model as well as the outcome model):

```
# doEffectDecomp.int is a user-written function
# See appendix for the function
doEffectDecomp.int(analytic.miss, ind = NULL, varlist = varlist)
#>          TE         DE         IE         PM
#> 1.54472021 1.48868864 1.03763821 0.08496674
# try bootstrap on it?
```

Visualization for main + interactions

```
set.seed(504)
bootresInt <- boot(data=analytic.miss, statistic=doEffectDecomp.int,
                R = 5, parallel = "multicore", ncpus=5,
                varlist = varlist)
```

R = 5 is not reliable for bootstrap. In real applications, try 250 at least.

```
bootci1i <- boot.ci(bootresInt,type = "perc",index=1)
#> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints
bootci2i <- boot.ci(bootresInt,type = "perc",index=2)
#> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints
bootci3i <- boot.ci(bootresInt,type = "perc",index=3)
#> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints
bootci4i <- boot.ci(bootresInt,type = "perc",index=4)
#> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints
```

```
bootresInt$R
#> [1] 5
# from saved boostrap results: bootresInt
```

```r
# (similar as before)
TEc <- c(bootresInt$t0[1], bootci1i$percent[4:5])
DEc <- c(bootresInt$t0[2], bootci2i$percent[4:5])
IEc <- c(bootresInt$t0[3], bootci3i$percent[4:5])
mat<- rbind(TEc,DEc,IEc)
colnames(mat) <- c("Point", "2.5%", "97.5%")
mat
#>        Point      2.5%    97.5%
#> TEc 1.544720 1.2931358 1.893575
#> DEc 1.488689 1.3040953 1.876521
#> IEc 1.037638 0.9742042 1.009088
plotCI(1:3, mat[,1], ui=mat[,3], li=mat[,2],
       xlab = "Estimates", ylab = "", xaxt="n")
axis(1, at=1:3,labels=c("TE","NDE","NIE"))
abline(h=1, lty = 2)
```

Appendix: OA-CVD Functions for bootstrap

These functions are written basically for performing bootstrap for the OA-CVD analysis. However, changing the covariates names/model-specifications should not be too hard, once you understand the basic steps.

```r
# without interactions (binary mediator)
doEffectDecomp
#> function (dat, ind = NULL, varlist)
#> {
#>     if (is.null(ind))
#>         ind <- 1:nrow(dat)
#>     d <- dat[ind, ]
#>     d$mediator <- ifelse(as.character(d$painmed) == "Yes", 1,
#>         0)
#>     d$exposure <- ifelse(as.character(d$OA) == "OA", 1, 0)
#>     d$outcome <- ifelse(as.character(d$CVD) == "event", 1, 0)
#>     d$exposureTemp <- d$exposure
#>     w.design0 <- svydesign(id = ~1, weights = ~weight, data = d)
#>     w.design <- subset(w.design0, miss == 0)
#>     fit.m <- svyglm(as.formula(paste0(paste0("mediator ~ exposureTemp  + "),
#>         paste0(varlist, collapse = "+"))), design = w.design,
#>         family = quasibinomial("logit"))
#>     d1 <- d2 <- d
#>     d1$exposure.counterfactual <- d1$exposure
#>     d2$exposure.counterfactual <- !d2$exposure
#>     newd <- rbind(d1, d2)
#>     newd <- newd[order(newd$ID), ]
#>     newd$exposureTemp <- newd$exposure
#>     w <- predict(fit.m, newdata = newd, type = "response")
#>     direct <- ifelse(newd$mediator, w, 1 - w)
#>     newd$exposureTemp <- newd$exposure.counterfactual
#>     w <- predict(fit.m, newdata = newd, type = "response")
#>     indirect <- ifelse(newd$mediator, w, 1 - w)
#>     newd$W.mediator <- indirect/direct
#>     summary(newd$W.mediator)
#>     newd$S.w <- with(newd, weight)
#>     summary(newd$S.w)
#>     newd$SM.w <- with(newd, (W.mediator * S.w))
#>     newd$SM.w[is.na(newd$SM.w)] <- 0
#>     summary(newd$SM.w)
#>     w.design0 <- svydesign(id = ~1, weights = ~SM.w, data = newd)
#>     w.design <- subset(w.design0, miss == 0)
#>     fit <- svyglm(as.formula(paste0(paste0("outcome ~ exposure + exposure.counterfactual +",
#>         paste0(varlist, collapse = "+"))), design = w.design,
#>         family = quasibinomial("logit"))
```

```
#>     TE <- exp(sum(coef(fit)[c("exposure", "exposure.counterfactual")]))
#>     DE <- exp(unname(coef(fit)["exposure"]))
#>     IE <- exp(unname(coef(fit)[c("exposure.counterfactual")]))
#>     PM <- log(IE)/log(TE)
#>     return(c(TE = TE, DE = DE, IE = IE, PM = PM))
#> }
#> <bytecode: 0x000002791991c048>

# with interactions (binary mediator)
doEffectDecomp.int
#> function (dat, ind = NULL, varlist)
#> {
#>     if (is.null(ind))
#>         ind <- 1:nrow(dat)
#>     d <- dat[ind, ]
#>     d$exposureTemp <- d$exposure
#>     w.design0 <- svydesign(id = ~1, weights = ~weight, data = d)
#>     w.design <- subset(w.design0, miss == 0)
#>     fit.m <- svyglm(as.formula(paste0(paste0("mediator ~ exposureTemp + phyact*diab +"),
#>         paste0(varlist, collapse = "+"))), design = w.design,
#>         family = quasibinomial("logit"))
#>     d1 <- d2 <- d
#>     d1$exposure.counterfactual <- d1$exposure
#>     d2$exposure.counterfactual <- !d2$exposure
#>     newd <- rbind(d1, d2)
#>     newd <- newd[order(newd$ID), ]
#>     newd$exposureTemp <- newd$exposure
#>     w <- predict(fit.m, newdata = newd, type = "response")
#>     direct <- ifelse(newd$mediator, w, 1 - w)
#>     newd$exposureTemp <- newd$exposure.counterfactual
#>     w <- predict(fit.m, newdata = newd, type = "response")
#>     indirect <- ifelse(newd$mediator, w, 1 - w)
#>     newd$W.mediator <- indirect/direct
#>     summary(newd$W.mediator)
#>     newd$S.w <- with(newd, weight)
#>     summary(newd$S.w)
#>     newd$SM.w <- with(newd, (W.mediator * S.w))
#>     newd$SM.w[is.na(newd$SM.w)] <- 0
#>     summary(newd$SM.w)
#>     w.design0 <- svydesign(id = ~1, weights = ~SM.w, data = newd)
```

```
#>     w.design <- subset(w.design0, miss == 0)
#>     fit <- svyglm(as.formula(paste0(paste0("outcome ~ exposure + exposure.counterfactual +",
#>         paste0(varlist, collapse = "+"))), design = w.design,
#>         family = quasibinomial("logit"))
#>     TE <- exp(sum(coef(fit)[c("exposure", "exposure.counterfactual")]))
#>     DE <- exp(unname(coef(fit)["exposure"]))
#>     IE <- exp(unname(coef(fit)[c("exposure.counterfactual")]))
#>     PM <- log(IE)/log(TE)
#>     return(c(TE = TE, DE = DE, IE = IE, PM = PM))
#> }
#> <bytecode: 0x0000027952a1b970>
```

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

R functions (I)

The list of new R functions introduced in this *mediation analysis* lab component are below:

Function_name	Package_name	Use
boot	boot	To conduct bootstrap resampling
boot.ci	boot	To calculate confidence intervals from bootstrap samples

Quiz (I)

> ⚠ Warning
>
> We are currently in the beta testing phase for our quiz page updates, so please be aware that some responses may not be accurate. Thank you for your patience.
> For those interested in downloading the quiz questions, you'll find a section below dedicated to downloading.

Live Quiz

You can click on the live quiz options to either see hints of why your response may be incorrect (response marked in red), or whether your choice of response is correct (response marked in green):

Download Quiz

1. **Downloading the File:**
   - Navigate to the link: See here.
   - Right-click on the link and select "Save link as…" from the dropdown menu.
   - Alternatively click here for downloading the quizzes.
   - Choose a destination folder on your computer where you'd like to save the file (e.g., Desktop). **Remember this location**, as you'll need to navigate to it later.
2. **Setting Up RStudio:**

- If you don't have RStudio installed, see the download link in here.
- Launch RStudio after installing.

3. **Installing Necessary R Packages:**

   - Before running the R Markdown file, ensure you have the required packages. In RStudio's console (located at the bottom left by default), enter the following commands to install them:

```r
install.packages("learnr")
install.packages("xfun")
```

4. **Opening and Running the File:**

   - In RStudio, go to `File` > `Open File...`.
   - Navigate to the folder where you saved the `Rmd` file and select it to open.
   - Once the file is open in RStudio, you'll see a "Run Document" button (green) at the top of the script editor. Click on it to run the R Markdown Quiz file.

Part XIV

Writing tools

Background

The tutorial offers a detailed guide on using R, RStudio, and GitHub for collaborative manuscript writing and management, covering aspects from software installation, through manuscript creation and editing, to publishing and sharing via GitHub Pages. Additionally, it introduces various supplementary resources, including LaTex, TablesGenerator, and Zotero, to further assist and streamline the research and writing processes.

> ❗ Important
>
> **Datasets**:
> All of the datasets used in this tutorial can be accessed from this GitHub repository folder

Overview of tutorials

Scientific Writing Collaboratively

In this tutorial, we navigated through a comprehensive guide on utilizing R, RStudio, and GitHub for collaborative manuscript writing. Beginning with the installation of R and RStudio, we explored creating a GitHub repository and managing it using GitHub Desktop. Subsequently, we delved into using RStudio to create a manuscript with a Bookdown template, compiling it, and updating the manuscript content. The tutorial also covered updating the GitHub repository with manuscript changes and publishing the manuscript using GitHub Pages, enabling a seamless, collaborative, and organized manuscript writing and sharing process, ensuring efficiency and ease in academic writing collaborations.

Formatting Resources

This section provides a wealth of tools and platforms beneficial for researchers and writers in managing and creating scientific

documents. It encompasses LaTex and ShareLaTeX for document preparation, TablesGenerator for converting tables from MS Word to various formats, and R packages like "officer" and "flextable" for generating tables and charts. It also introduces draw.io for crafting flow charts, platforms like jane for identifying suitable journals, officetimeline for creating Gantt charts, and Google Docs for real-time collaborative writing. Moreover, it highlights Zotero and ZoteroBib as comprehensive tools for reference management and bibliography creation, facilitating organized, collaborative, and streamlined research and writing processes.

Tip

Optional Content:
You'll find that some sections conclude with an optional video walkthrough that demonstrates the code. Keep in mind that the content might have been updated since these videos were recorded. Watching these videos is optional.

Warning

Bug Report:
Fill out this form to report any issues with the tutorial.

Collaborative Writing

This tutorial provides a step-by-step guide to using R, RStudio, GitHub, and GitHub Desktop for collaborative manuscript writing. These tools facilitate collaboration and help keep track of the manuscript writing progress.

All necessary files for this tutorial are available here. If you're new to Rmarkdown, consider revisiting the introductory tutorial.

GitHub Account

- Visit GitHub and click "Sign up", and follow the instructions to create your account.
- On GitHub, click the "+" or "new" icon on the top right and select "New repository".
- Name your repository, add a description, initialize it with a README, and click "Create repository".

Necessary Software

- R and RStudio (revisit previous chapter)
- GitHub Desktop. Install and sign in with your GitHub credentials.

Cloning Repository

- On GitHub Desktop, go to "File" > "Clone repository",
- choose the repository you want to clone and select the local path,
- and click "Clone".
- Navigate to the local path and ensure all files are cloned.

Bookdown Template in RStudio

- In RStudio, go to "File" > "New Project" > "New Directory" > "Book Project using bookdown".
- Name your project (e.g., "test1") and create it.
- Navigate to the project directory, copy all files, and paste them into your original repository folder.
- Rename the project file to a suitable name (e.g., "template").

Compiling

- Open the index file in your RStudio project.
- Update the YAML header and chapter/section names as per your manuscript.
- Compile the document into HTML and/or PDF using the "Build" tab in RStudio.

Updating GitHub Repository

- Make changes to your manuscript files in RStudio.
- Open GitHub Desktop, review changes, commit them with a descriptive message, and push to the origin.

Publishing Using GitHub Pages

- In your GitHub repository, create a new folder named "docs" and copy your compiled HTML files into it.
- Go to the settings of your GitHub repository, navigate to "GitHub Pages", and set the source to the "docs" folder.
- Your manuscript will be available at "username.github.io/repository_name".

Invite collaborators

To invite collaborators to a GitHub repository, begin by navigating to your desired repository and clicking on the "Settings" tab. In the left sidebar, select "Manage Access" and click "Invite a collaborator." Enter the username, full name, or email

of the person you wish to invite and select them from the suggestions. Assign a role to define their access level and send the invitation. The invitee can accept via the email link sent or directly through their GitHub account. You can review and manage all collaborators and pending invitations in the "Manage Access" section.

Video content (optional)

> **Tip**
>
> For those who prefer a video walkthrough, feel free to watch the video below, which offers a description of an earlier version of the above content.

Useful resources

- Manuscript template. The output website can be accessed here

Formatting Tools

The tutorial elucidates a variety of tools and methodologies aimed at streamlining and enhancing academic writing and presentation creation, discussing topics such as utilizing a typesetting system, converting tables across different formats, employing various R packages for enhanced data visualization and presentation, drawing flow and Gantt charts, crafting HTML5 presentations, enabling collaborative writing and document sharing, managing references efficiently, formatting articles, and employing specific platforms for identifying appropriate journals for publishing academic articles.

LaTex

LaTeX is a high-quality software system for typesetting document. LaTeX is designed for the production of technical and scientific documentation, such as book, articles.

Get an account in ShareLaTeX/Overleaf.

Table conversion

From MS word to latex / markdown / HTML in TablesGenerator.

You can use `tableone` package to generate csv file, and then import them in the TablesGenerator to convert to HTML to paste to doc file!

```
tab1x <- print(tab1, quote = FALSE, noSpaces = TRUE, printToggle = FALSE)
write.csv(tab1x, file = "tab1x.csv")
```

Fancy table and chart generators:

R Packages

- officer
- officedown
- flextable
- mschart

Drawing flow chart

draw.io

Presentation

The "xaringan" package, derived from a love for the Japanese manga and anime "Naruto", serves as an R Markdown extension, and it facilitates the creation of distinctively styled HTML5 presentations by leveraging the JavaScript library remark.js. Originating from an intent to produce a unique, though not widely adopted style, due to its potentially challenging pronunciation unless familiar with the anime, "xaringan" offers significant customizability in presentation design and has garnered additional theme contributions from its user community. Despite only supporting Markdown, "xaringan" enhances remark.js by introducing support for R Markdown and additional utilities, simplifying the slide-building and previewing processes. Further insights into "xaringan", its background, and its utility can be explored through its documentation.

Gantt charts

officetimeline

Simultaneous collaborative writing

Google Docs offers a platform for real-time collaborative writing. Multiple users can edit documents simultaneously, and changes are saved automatically. Note that Google Docs requires sign-in for a Google account.

- Commenting and Suggesting: Use the comment and suggest features to provide feedback without altering the original text.
- Revision History: Navigate through the revision history to view changes and revert to previous versions if needed.
- Sharing and Permissions: Manage who can view, comment, or edit the document with varied permission levels.

Reference manager

Zotero stands out as a free, open-source reference management software that assists researchers, academics, and students in organizing, managing, and formatting their citations and bibliographies. It's not just a reference manager but also a powerful tool for collaborative work on research projects. Try the Zotero desktop manager as well for assisting with reference inserting.

Zotero syncs data across devices, ensuring that users can access their libraries from any location. Users can work offline with Zotero, and any changes made will be synchronized when the internet connection is restored.

ZoteroBib: Use ZoteroBib to generate bibliographies instantly without creating an account or installing software.

Article formatting

The `rticles` package in R provides a diverse selection of templates for creating academic articles and is easily accessible directly within the RStudio environment by navigating through **File -> New File -> R Markdown**, where users can select their desired template. For users not utilizing RStudio, the installation of Pandoc is requisite, with articles being creatable using the `rmarkdown::draft()` function, and specifying

the template and package parameters as needed. Additionally, the package enables viewing a list of available journal names using `rticles::journals()`. To employ enhanced features, such as automatic figure numbering and cross-referencing of tables, users can utilize functionalities from the `bookdown` package. This involves adjusting the YAML to use `bookdown::pdf_book` as the output format, and designating the chosen `rticles` template as the `base_format`. Comprehensive details and tutorials regarding the use of the `rticles` package can be found in its online documentation. The complete array of options can be explored within the R Markdown templates window in RStudio, via the packages's GitHub readme or accessed programmatically via the following function:

```r
rticles::journals()
#>  [1] "acm"            "acs"            "aea"            "agu"
#>  [5] "ajs"            "amq"            "ams"            "arxiv"
#>  [9] "asa"            "bioinformatics" "biometrics"     "copernicus"
#> [13] "ctex"           "elsevier"       "frontiers"      "glossa"
#> [17] "ieee"           "ims"            "informs"        "iop"
#> [21] "isba"           "jasa"           "jedm"           "joss"
#> [25] "jss"            "lipics"         "lncs"           "mdpi"
#> [29] "mnras"          "oup_v0"         "oup_v1"         "peerj"
#> [33] "pihph"          "plos"           "pnas"           "rjournal"
#> [37] "rsos"           "rss"            "sage"           "sim"
#> [41] "springer"       "tf"             "trb"            "wellcomeor"
```

Find appropriate journals

- jane
- See more extensive list here
- Specific Epidemiology-focus journals

Summary

Category	Tool	Description
Academic Search Engine	Google Scholar	Freely accessible search engine indexing scholarly literature.
Article Formatting	rticles (R Package)	R package providing templates for various academic journals.
Brainstorming & Mapping	MindMeister	Online mind mapping tool for brainstorming and collaborative visualization.
Document Creation & Editing	Overleaf	Collaborative LaTeX editor online.
	Google Docs	Platform for real-time collaborative writing.
	officer (R Package)	R package for generating Word and PowerPoint files.
	officedown (R Package)	R package to produce Word documents with officer.
Gantt Chart Generators	Office Timeline	Online tool for creating Gantt charts.
Journal Finding	jane	Tool to assist in finding the right journal for publishing.

Category	Tool	Description
	Custom List	Extensive list and guide on finding suitable journals.
	Epidemiology Journals	Specific list of epidemiology-focus journals.
Note & Research Management	Evernote	Note-taking and organization tool for managing research notes and drafts.
Presentation & Sharing	Prezi	Dynamic and visually engaging presentation creation tool.
	SlideShare	Platform for sharing presentations and professional documents.
Presentation	xaringan (R Package)	R Markdown extension for creating presentations using remark.js.
Project Management	Asana	Project management tool for workflow organization and collaboration.
	Trello	Project management tool for task tracking and collaboration.

Category	Tool	Description
Reference Management	EndNote	Reference management software for organizing and integrating references.
	JabRef	Open-source bibliography reference manager using BibTeX.
	Mendeley	Reference management and academic social network.
	Paperpile	Reference management and academic research library.
	Zotero	Reference management and collaborative tool.
	ZoteroBib	Quick bibliography generation tool.
Research Identity Management	ORCID	Provides a persistent digital identifier to distinguish researchers.
Table & Chart Generators	draw.io	Online diagram software for flow charts and various diagrams.
	TablesGenerator	Converts tables to LaTeX, markdown, HTML formats.

Category	Tool	Description
	flextable (R Package)	R package for tabular reporting in various formats (Word, HTML, etc.).
	mschart (R Package)	R package to create PowerPoint charts.
Writing & Editing	Authorea	Collaborative platform for writing, citing, and publishing.
	Grammarly	Writing assistant for grammar and style enhancement.

Archer, K. J., and S. Lemeshow. 2006. "Goodness-of-Fit Test for a Logistic Regression Model Fitted Using Survey Sample Data." *The Stata Journal* 6 (1): 97–105.

Austin, Peter C. 2010. "Absolute Risk Reductions, Relative Risks, Relative Risk Reductions, and Numbers Needed to Treat Can Be Obtained from a Logistic Regression Model." *Journal of Clinical Epidemiology* 63 (1): 2–6.

Austin, Peter C. 2011. "A Tutorial and Case Study in Propensity Score Analysis: An Application to Estimating the Effect of in-Hospital Smoking Cessation Counseling on Mortality." *Multivariate Behavioral Research* 46 (1): 119–51.

Austin, Peter C, Nathaniel Jembere, and Maria Chiu. 2018. "Propensity Score Matching and Complex Surveys." *Statistical Methods in Medical Research* 27 (4): 1240–57.

Baron, Reuben M, and David A Kenny. 1986. "The Moderator–Mediator Variable Distinction in Social Psychological Research: Conceptual, Strategic, and Statistical Considerations." *Journal of Personality and Social Psychology* 51 (6): 1173.

Bi, Qifang, Katherine E Goodman, Joshua Kaminsky, and Justin Lessler. 2019. "What Is Machine Learning? A Primer for the Epidemiologist." *American Journal of Epidemiology* 188 (12): 2222–39.

Bieler, Gayle S, G Gordon Brown, Rick L Williams, and Donna

J Brogan. 2010. "Estimating Model-Adjusted Risks, Risk Differences, and Risk Ratios from Complex Survey Data." *American Journal of Epidemiology* 171 (5): 618–23.

Bilder, Christopher R, and Thomas M Loughin. 2014. *Analysis of Categorical Data with r*. CRC Press.

Bondarenko, Vadim, and FI Consulting. 2023. "The Bootstrap Approach to Managing Model Uncertainty." https://rstudio-pubs-static.s3.amazonaws.com/90467_c70206f3dc864d53bf36072207ee011d.html.

Bruin, J. 2023. "Advanced Topics in Survey Data Analysis." 2023. https://stats.oarc.ucla.edu/other/mult-pkg/seminars/advanced-topics-in-survey-data-analysis/.

Buuren, S van, and Karin Groothuis-Oudshoorn. 2010. "Mice: Multivariate Imputation by Chained Equations in r." *Journal of Statistical Software*, 1–68.

Canada, Statistics. 2005. "Canadian Community Health Survey (CCHS), Cycle 3.1." Author Ottawa.

CDC. 2023. "NHANES Web Tutorial Frequently Asked Questions (FAQs)." https://wwwn.cdc.gov/nchs/nhanes/continuousnhanes/faq.aspx.

CDC,NCHS. 2023. "National Health and Nutrition Examination Survey Data." https://wwwn.cdc.gov/nchs/nhanes/.

Christodoulou, Evangelia, Jie Ma, Gary S Collins, Ewout W Steyerberg, Jan Y Verbakel, and Ben Van Calster. 2019. "A Systematic Review Shows No Performance Benefit of Machine Learning over Logistic Regression for Clinical Prediction Models." *Journal of Clinical Epidemiology* 110: 12–22.

Cole, Stephen R, and Miguel A Hernán. 2008. "Constructing Inverse Probability Weights for Marginal Structural Models." *American Journal of Epidemiology* 168 (6): 656–64.

Connors, Alfred F, Theodore Speroff, Neal V Dawson, Charles Thomas, Frank E Harrell, Douglas Wagner, Norman Desbiens, et al. 1996. "The Effectiveness of Right Heart Catheterization in the Initial Care of Critically Ill Patients." *Jama* 276 (11): 889–97. https://tinyurl.com/Connors1996.

Coxe, Stefany, Stephen G. West, and Leona S. Aiken. 2009. "The Analysis of Count Data: A Gentle Introduction to Poisson Regression and Its Alternatives." *Journal of Personality Assessment* 91 (2): 121–36. https://doi.org/10.

1080/00223890802634175.

Cui, Jianwen, and Guoqing Qian. 2007. "Selection of Working Correlation Structure and Best Model in GEE Analyses of Longitudinal Data." *Communications in Statistics—Simulation and Computation®* 36 (5): 987–96.

Dhana, A. 2023. "R & Python for Data Science." https://datascienceplus.com/.

DuGoff, Eva H, Megan Schuler, and Elizabeth A Stuart. 2014. "Generalizing Observational Study Results: Applying Propensity Score Methods to Complex Surveys." *Health Services Research* 49 (1): 284–303.

Etminan, Mahyar, Gary S Collins, and Mohammad Ali Mansournia. 2020. "Using Causal Diagrams to Improve the Design and Interpretation of Medical Research." *Chest* 158 (1): S21–28.

Faraway, Julian J. 2016. *Extending the linear model with R: generalized linear, mixed effects and nonparametric regression models*. CRC press.

Flegal, Katherine M, Deanna Kruszon-Moran, Margaret D Carroll, Cheryl D Fryar, and Cynthia L Ogden. 2016. "Trends in Obesity Among Adults in the United States, 2005 to 2014." *Jama* 315 (21): 2284–91.

Frank, Hanna A, and Mohammad Ehsanul Karim. 2023. "Implementing TMLE in the Presence of a Continuous Outcome." *Research Methods in Medicine & Health Sciences*, 26320843231176662.

Granger, Elizabeth, Jamie C. Sergeant, and Mark Lunt. 2019. "Avoiding Pitfalls When Combining Multiple Imputation and Propensity Scores." *Statistics in Medicine* 38 (26): 5120–32.

Greenland, Sander. 2004. "Model-Based Estimation of Relative Risks and Other Epidemiologic Measures in Studies of Common Outcomes and in Case-Control Studies." *American Journal of Epidemiology* 160 (4): 301–5.

Greenland, Sander, and Neil Pearce. 2015. "Statistical Foundations for Model-Based Adjustments." *Annual Review of Public Health* 36: 89–108. https://doi.org/10.1146/annurev-publhealth-031914-122531.

Greenland, Sander, Judea Pearl, and James M Robins. 1999a. "Causal Diagrams for Epidemiologic Research." *Epidemiology*, 37–48.

———. 1999b. "Confounding and Collapsibility in Causal Inference." *Statistical Science* 14 (1): 29–46.

Gruber, Susan, and Mark J van der Laan. 2010. "A Targeted Maximum Likelihood Estimator of a Causal Effect on a Bounded Continuous Outcome." *The International Journal of Biostatistics* 6 (1).

Gruber, S., M. Van Der Laan, and C. Kennedy. 2020. *Package 'Tmle'*. https://cran.r-project.org/web/packages/tmle/tmle.pdf.

Hawkins, Douglas M. 1981. "A New Test for Multivariate Normality and Homoscedasticity." *Technometrics* 23 (1): 105–10.

Heeringa, Steven G., Brady T. West, and Patricia A. Berglund. 2014. "Regression with Complex Samples." In *The SAGE Handbook of Regression Analysis and Causal Inference*, edited by Henning Best and Christof Wolf. SAGE Publications.

Heeringa, Steven G, Brady T West, and Patricia A Berglund. 2017. *Applied Survey Data Analysis*. Chapman; Hall/CRC.

Heinze, Georg, Christine Wallisch, and Daniela Dunkler. 2018. "Variable Selection–a Review and Recommendations for the Practicing Statistician." *Biometrical Journal* 60 (3): 431–49.

Hernán, Miguel A., and James M. Robins. 2020. "Chapter 3." In *Causal Inference: What If*. Boca Raton: Chapman & Hall/CRC.

Hernán, Miguel A, and James M Robins. 2006. "Estimating Causal Effects from Epidemiological Data." *Journal of Epidemiology and Community Health* 60 (7): 578.

Hiemstra, Bart, Frederik Keus, Jørn Wetterslev, Christian Gluud, and Iwan CC van der Horst. 2019. "DEBATE-Statistical Analysis Plans for Observational Studies." *BMC Medical Research Methodology* 19 (1): 1–10.

Hosmer, Jr., David W., Stanley Lemeshow, and Rodney X. Sturdivant. 2013. *Applied Logistic Regression, 3rd Edition*. Hoboken, NJ: John Wiley & Sons.

Hossain, Md Belal, Jacek A Kopec, Mohammad Atiquzzaman, and Mohammad Ehsanul Karim. 2022. "The Association Between Rheumatoid Arthritis and Cardiovascular Disease Among Adults in the United States During 1999–2018, and

Age-Related Effect Modification in Relative and Absolute Scales." *Annals of Epidemiology* 71: 23–30.

Hothorn, Torsten, and Brian S Everitt. 2014. *A Handbook of Statistical Analyses Using r*. CRC press.

Hughes, Rachael A., Jon Heron, Jonathan A. Sterne, and Kate Tilling. 2019. "Accounting for Missing Data in Statistical Analyses: Multiple Imputation Is Not Always the Answer." *International Journal of Epidemiology* 1: 11.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning*. Vol. 112. Springer.

Jamshidian, Mortaza, and Siavash Jalal. 2010. "Tests of Homoscedasticity, Normality, and Missing Completely at Random for Incomplete Multivariate Data." *Psychometrika* 75 (4): 649–74.

Kahan, Brennan C, Tahania Ahmad, Gordon Forbes, and Suzie Cro. 2020. "Public Availability and Adherence to Prespecified Statistical Analysis Approaches Was Low in Published Randomized Trials." *Journal of Clinical Epidemiology* 128: 29–34.

Karim, Ehsan. 2021. "Understanding Basics and Usage of Machine Learning in Medical Literature." 2021. https://ehsanx.github.io/into2ML/.

———. 2023. "Case Study 2: Risk of Cardiovascular Disease Among Osteoarthritis Patients." https://ssc.ca/en/case-study/case-study-2-risk-cardiovascular-disease-among-osteoarthritis-patients.

Karim, Ehsan, and Hanna Frank. 2021. "ehsanx/TMLEworkshop: R Guide for TMLE in Medical Research." Zenodo. https://doi.org/10.5281/zenodo.5246085.

Karim, ME. 2021. "Understanding Propensity Score Matching." 2021. https://ehsanx.github.io/psw/.

Karim, Mohammad Ehsanul, Helen Tremlett, Feng Zhu, John Petkau, and Elaine Kingwell. 2021. "Dealing with Treatment-Confounder Feedback and Sparse Follow-up in Longitudinal Studies: Application of a Marginal Structural Model in a Multiple Sclerosis Cohort." *American Journal of Epidemiology* 190 (5): 908–17.

KDnuggets. 2023. "Dataset Splitting Best Practices in Python." https://www.kdnuggets.com/2020/05/dataset-splitting-best-practices-python.html.

Kleinbaum, David G., and Mitchel Klein. 2011. *Survival Analysis: A Self-Learning Text, Third Edition*. New York, NY: Springer.

Kleinman, Lawrence C, and Edward C Norton. 2009. "What's the Risk? A Simple Approach for Estimating Adjusted Risk Measures from Nonlinear Models Including Logistic Regression." *Health Services Research* 44 (1): 288–302.

Koch, G. G., D. H. Freeman Jr, and J. L. Freeman. 1975. "Strategies in the Multivariate Analysis of Data from Complex Surveys." *International Statistical Review/Revue Internationale de Statistique*, 59–78.

Kuhn, Max. 2023a. "Available Models." https://topepo.github.io/caret/available-models.html.

———. 2023b. "Data Splitting." https://topepo.github.io/caret/data-splitting.html.

———. 2023c. "Model Training and Tuning." https://topepo.github.io/caret/model-training-and-tuning.html.

Kuhn, Max, and Kjell Johnson. 2013. *Applied Predictive Modeling*. Vol. 26. Springer.

Kuhn, Max, Kjell Johnson, Max Kuhn, and Kjell Johnson. 2013. "Over-Fitting and Model Tuning." *Applied Predictive Modeling*, 61–92.

Lange, Theis, Kristoffer W Hansen, Rune Sørensen, and Søren Galatius. 2017. "Applied Mediation Analyses: A Review and Tutorial." *Epidemiology and Health* 39.

Lederer, David J, Scott C Bell, Richard D Branson, James D Chalmers, Rebecca Marshall, David M Maslove, Peter W Stewart, et al. 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.

Leite, Walter L., Laura M. Stapleton, and Eduardo F. Bettini. 2018. "Propensity Score Analysis of Complex Survey Data with Structural Equation Modeling: A Tutorial with Mplus." *Structural Equation Modeling: A Multidisciplinary Journal*, 1–22.

Lenis, Diego, Benjamin Ackerman, and Elizabeth A. Stuart. 2018. "Measuring Model Misspecification: Application to Propensity Score Methods with Complex Survey Data." *Computational Statistics & Data Analysis*.

Lenis, Diego, Thuan Quoc Nguyen, Dong, and Elizabeth A. Stuart. 2017. "It's All about Balance: Propensity Score Matching in the Context of Complex Survey Data." *Biostatistics*.

Li, Meng, Shoumeng Yan, Xing Li, Shan Jiang, Xiaoyu Ma, Hantong Zhao, Jiagen Li, et al. 2020. "Association Between Blood Pressure and Dietary Intakes of Sodium and Potassium Among US Adults Using Quantile Regression Analysis NHANES 2007–2014." *Journal of Human Hypertension* 34 (5): 346–54.

Little, RJA, and DB Rubin. 1987. "Multiple Imputation for Nonresponse in Surveys." *John Wiley & Sons, Inc.. Doi* 10: 9780470316696.

Little, Roderick JA. 1988. "A Test of Missing Completely at Random for Multivariate Data with Missing Values." *Journal of the American Statistical Association* 83 (404): 1198–1202.

Liu, Yun, Po-Hsuan Cameron Chen, Jonathan Krause, and Lily Peng. 2019. "How to Read Articles That Use Machine Learning: Users' Guides to the Medical Literature." *Jama* 322 (18): 1806–16.

Luijken, Kim, Rolf HH Groenwold, Maarten van Smeden, Susanne Strohmaier, and Georg Heinze. 2022. "A Comparison of Full Model Specification and Backward Elimination of Potential Confounders When Estimating Marginal and Conditional Causal Effects on Binary Outcomes from Observational Data." *Biometrical Journal*.

Lumley, Thomas. 2011. *Complex Surveys: A Guide to Analysis Using r*. Vol. 565. John Wiley & Sons.

———. 2017. "Pseudo-R2 Statistics Under Complex Sampling." *Australian & New Zealand Journal of Statistics* 59 (2): 187–94.

Lumley, Thomas, and Alan Scott. 2014. "Tests for Regression Models Fitted to Survey Data." *Australian & New Zealand Journal of Statistics* 56 (1): 1–14.

———. 2015. "AIC and BIC for Modeling with Complex Survey Data." *Journal of Survey Statistics and Methodology* 3 (1): 1–18.

Luque-Fernandez, Miguel Angel, Michael Schomaker, Bernard Rachet, and Mireille E Schnitzer. 2018. "Targeted Maximum Likelihood Estimation for a Binary Treatment: A

Tutorial." *Statistics in Medicine* 37 (16): 2530–46.

Mansournia, Mohammad Ali, and Sander Greenland. 2015. "The Relation of Collapsibility and Confounding to Faithfulness and Stability." *Epidemiology* 26 (4): 466–72.

Mullen, Katharine M., and Ivo H. M. van Stokkum. 2023. *Package 'Nnls'*. https://cran.r-project.org/web/packages/nnls/nnls.pdf.

Muller, Clemma J, and Richard F MacLehose. 2014. "Estimating Predicted Probabilities from Logistic Regression: Different Methods Correspond to Different Target Populations." *International Journal of Epidemiology* 43 (3): 962–70.

Naimi, Ashley I, and Brian W Whitcomb. 2020. "Estimating Risk Ratios and Risk Differences Using Regression." *American Journal of Epidemiology* 189 (6): 508–10.

Palis, Heather, Kirsten Marchand, and Eugenia Oviedo-Joekes. 2020. "The Relationship Between Sense of Community Belonging and Self-Rated Mental Health Among Canadians with Mental or Substance Use Disorders." *Journal of Mental Health* 29 (2): 168–75.

Peters, Junenette L, M Patricia Fabian, and Jonathan I Levy. 2014. "Combined Impact of Lead, Cadmium, Polychlorinated Biphenyls and Non-Chemical Risk Factors on Blood Pressure in NHANES." *Environmental Research* 132: 93–99.

Phillips, Rachael V., Mark J. van der Laan, Hana Lee, and Susan Gruber. 2023. "Practical Considerations for Specifying a Super Learner." *International Journal of Epidemiology* 52: 1276–85. https://doi.org/10.1093/ije/dyad023.

Polley, Eric. 2021. *Package 'SuperLearner'*. https://cran.r-project.org/web/packages/SuperLearner/SuperLearner.pdf.

Quick-R. 2023. "Importing Data." https://www.statmethods.net/input/importingdata.html.

Rahman, M Mushfiqur, Jacek A Kopec, Jolanda Cibere, Charlie H Goldsmith, and Aslam H Anis. 2013. "The Relationship Between Osteoarthritis and Cardiovascular Disease in a Population Health Survey: A Cross-Sectional Study." *BMJ Open* 3 (5): e002624.

Rao, J. N. K., and A. J. Scott. 1984. "On Chi-Squared Tests for Multiway Contingency Tables with Cell Proportions Estimated from Survey Data." *The Annals of Statistics*, 46–

60.

Raschka, Sebastian. 2023. "Bootstrap_point632_score: The .632 and .632+ Boostrap for Classifier Evaluation." 2023. https://rasbt.github.io/mlxtend/user_guide/evaluate/bootstrap_point632_score/.

RDocumentation. 2023. "Matchit: Matchit: Matching Software for Causal Inference." https://www.rdocumentation.org/packages/MatchIt/versions/1.0-1/topics/matchit.

rdrr. 2023. "Na.test: Little's Missing Completely at Random (MCAR) Test." https://rdrr.io/cran/misty/man/na.test.html.

Ridgeway, Greg, Stephanie A. Kovalchik, Beth Ann Griffin, and Mohammed U. Kabeto. 2015. "Propensity Score Analysis with Survey Weighted Data." *Journal of Causal Inference* 3 (2): 237–49.

Rochon, Justine, Andreas du Bois, and Theis Lange. 2014. "Mediation Analysis of the Relationship Between Institutional Research Activity and Patient Survival." *BMC Medical Research Methodology* 14 (1): 9.

Simoneau, Gabrielle, Fabio Pellegrini, Thomas PA Debray, Julie Rouette, Johanna Muñoz, Robert W Platt, John Petkau, et al. 2022. "Recommendations for the Use of Propensity Score Methods in Multiple Sclerosis Research." *Multiple Sclerosis Journal* 28 (9): 1467–80.

Stata. 2023. "How Can i Analyze a Subpopulation of My Survey Data in Stata?" https://stats.oarc.ucla.edu/stata/faq/how-can-i-analyze-a-subpopulation-of-my-survey-data-in-stata/.

Sterne, Jonathan A., and et al. 2009. "Multiple Imputation for Missing Data in Epidemiological and Clinical Research: Potential and Pitfalls." *BMJ* 338: b2393.

Steyerberg, Ewout W. 2019. *Clinical Prediction Models: A Practical Approach to Development, Validation, and Updating*. Vol. 2. Springer.

Steyerberg, Ewout W, Frank E Harrell Jr, Gerard JJM Borsboom, MJC Eijkemans, Yvonne Vergouwe, and J Dik F Habbema. 2001. "Internal Validation of Predictive Models: Efficiency of Some Procedures for Logistic Regression Analysis." *Journal of Clinical Epidemiology* 54 (8): 774–81.

Steyerberg, Ewout W, and Ewout W Steyerberg. 2019. "Over-

fitting and Optimism in Prediction Models." *Clinical Prediction Models: A Practical Approach to Development, Validation, and Updating*, 95–112.

Steyerberg, Ewout W, and Yvonne Vergouwe. 2014. "Towards Better Clinical Prediction Models: Seven Steps for Development and an ABCD for Validation." *European Heart Journal* 35 (29): 1925–31.

Steyerberg, Ewout W, Andrew J Vickers, Nancy R Cook, Thomas Gerds, Mithat Gonen, Nancy Obuchowski, Michael J Pencina, and Michael W Kattan. 2010. "Assessing the Performance of Prediction Models: A Framework for Some Traditional and Novel Measures." *Epidemiology (Cambridge, Mass.)* 21 (1): 128.

Stuart, Elizabeth A. 2018. "Chapter 28. Propensity Scores and Matching Methods." In *The Reviewer's Guide to Quantitative Methods in the Social Sciences, Second Edition*, edited by Gregory R. Hancock, Ralph O. Mueller, and Laura M. Stapleton. Routledge.

Susmann, Herb. 2016. *RNHANES: Facilitates Analysis of CDC NHANES Data*. https://CRAN.R-project.org/package=RNHANES.

Tennant, Paul W, Elizabeth J Murray, Kathryn F Arnold, Leigh Berrie, Matthew P Fox, Samantha C Gadd, and George TH Ellison. 2021. "Use of Directed Acyclic Graphs (DAGs) to Identify Confounders in Applied Health Research: Review and Recommendations." *International Journal of Epidemiology* 50 (2): 620–32.

Thabane, Lehana, Tara Thomas, Chenglin Ye, and James Paul. 2009. "Posing the Research Question: Not so Simple." *Canadian Journal of Anesthesia/Journal Canadien d'anesthésie* 56 (1): 71–79.

Thomas, D. R., and J. N. K. Rao. 1987. "Small-Sample Comparisons of Level and Power for Simple Goodness-of-Fit Statistics Under Cluster Sampling." *Journal of the American Statistical Association* 82 (398): 630–36.

Van Buuren, Stef. 2018b. *Flexible Imputation of Missing Data*. Chapman; Hall/CRC.

———. 2018a. *Flexible Imputation of Missing Data*. Chapman; Hall/CRC.

Van Buuren, Stef, and Karin Groothuis-Oudshoorn. 2011. "Mice: Multivariate Imputation by Chained Equations in

r." *Journal of Statistical Software* 45: 1–67.

VanderWeele, Tyler J. 2009. "On the Distinction Between Interaction and Effect Modification." *Epidemiology*, 863–71.

———. 2019. "Principles of Confounder Selection." *European Journal of Epidemiology* 34 (3): 211–19.

Vittinghoff, Eric, David V. Glidden, Stephen C. Shiboski, and Charles E. McCulloch. 2011a. "Predictor Selection." In *Regression Methods in Biostatistics: Linear, Logistic, Survival, and Repeated Measures Models*. Springer.

Vittinghoff, Eric, David V Glidden, Stephen C Shiboski, and Charles E McCulloch. 2011b. *Regression Methods in Biostatistics: Linear, Logistic, Survival, and Repeated Measures Models*. Springer Science & Business Media.

Vittinghoff, Eric, David V Glidden, Stephen C Shiboski, Charles E McCulloch, Eric Vittinghoff, David V Glidden, Stephen C Shiboski, and Charles E McCulloch. 2012. "Predictor Selection." *Regression Methods in Biostatistics: Linear, Logistic, Survival, and Repeated Measures Models*, 395–429.

Westreich, Daniel, and Sander Greenland. 2013. "The Table 2 Fallacy: Presenting and Interpreting Confounder and Modifier Coefficients." *American Journal of Epidemiology* 177 (4): 292–98.

Wikipedia. 2023a. "Coefficient of Determination." https://en.wikipedia.org/wiki/Coefficient_of_determination.

———. 2023b. "Cross-Validation (Statistics)." https://en.wikipedia.org/wiki/Cross-validation_(statistics).

———. 2023c. "Elbow Method (Clustering)." https://en.wikipedia.org/wiki/Elbow_method_(clustering).

———. 2023d. "One-Way Analysis of Variance." https://en.wikipedia.org/wiki/One-way_analysis_of_variance.

Williamson, Elizabeth J, Alex J Walker, Krishnan Bhaskaran, Seb Bacon, Chris Bates, Caroline E Morton, Helen J Curtis, et al. 2020. "Factors Associated with COVID-19-Related Death Using OpenSAFELY." *Nature* 584 (7821): 430–36.

Wright, Sewall. 1921. "Correlation and Causation." *J. Agric. Res* 20: 557–80.

Zanutto, Elaine L. 2006. "A Comparison of Propensity Score and Linear Regression Analysis of Complex Survey Data." *Journal of Data Science* 4 (1): 67–91.